Two mathematical security aspects of the RSA cryptosystem: signature padding schemes and key generation with a backdoor

Geneviève Arboit

Doctorate of Philosophy

School of Computer Science

McGill University

Montreal, Quebec

February 2008

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of philosophy doctorate in computer science.

DEDICATION

This document is dedicated to all people dear to me.

ACKNOWLEDGEMENTS

Firstly, I thank my thesis adviser, Claude Crépeau, for suggesting the topic of backdoors and for his counsel. I also thank Jean-Marc Robert and Alain Slakmon for their precious reading and comments.

I thank my parents, my family, my friends, and Dan Nyborg who put up with my excentric notion of becoming a doctor. I thank the crypto and quantum info lab people, as well as Geza Szamosi, Faith Fich, Charles Rackoff, Laurie Hendren, David Avis, Prakash Panangaden, and Bettina Kemme, for their interest and disinterested help. In particular, I owe thanks to Isabelle Déchène, Raza Ali Kazmi, Simon Pierre Desrosiers, Martin Courchesne, Natasa Przulj, Nicole Witen (came up with the Casanova story), Valerie Hongoh, Jenifer Thorsley, Shashikala Sivapragasam, Paul Dumais, Hugo Touchette, and François Pitt. Concerning publications, I would additionally like to thank David Naccache, Jean-Sébastien Coron, and a few anonymous referees for useful comments.

The McGill kendo club and d'Orangeville sensei are a late addition to the people I acknowledge the help of and deeply thank, for making the writing of a philosophy doctorate thesis appear to be *relatively* easy.

ABSTRACT

This work presents mathematical properties of the RSA cryptosystem. The topics of backdoors and padding algorithm are developped.

For padding schemes, we give a practical instantiation with a security reduction. It is based on the compression function of SHA-1 without any chaining function. Our solution has the advantage over the previous one of removing the relation of the output length of the compression function to the length of the RSA modulus.

For backdoors, improvements on definitions, existing algorithms as well as extensions of existing theorems are shown. The definitions pertaining to backdoored key generators are improved as to make their analysis uniform and comparable. New algorithms are presented and compared to existing ones as to show improvements mainly on their running time, the indistinguishability of the keys produced, and that some of these new algorithms are, for all practical purposes, the best that may be called asymmetric. Our theorem on the correctness (or *completeness*) of one of our better backdoored key generators is a generalization of a theorem of Boneh, Durfee and Frankel's on partial information on the decryption exponent.

RÉSUMÉ

Ce travail présente des propriétés mathématiques du cryptosystème RSA. Les sous-domaines développés sont ceux des portes dérobées et des algorithmes de remplissage.

Pour les algorithmes de remplissage, nous présentons une instantiation pratique avec une réduction de sécurité. Elle est basée sur la fonction de compression de SHA-1 sans user d'aucune fonction d'enchaînement. Notre solution a l'avantage, par rapport à la précédente, d'enlever la relation entre la longueur de sortie de la fonction de compression et la longueur du modulo RSA.

Pour les portes dérobées, des améliorations sur les définitions, les algorithmes existants aussi bien que des prolongements des théorèmes existants sont exposés. Les définitions concernant les générateurs de clés à porte dérobée sont améliorées afin que leur analyse soit uniforme et comparable. De nouveaux algorithmes sont présentés et comparés à ceux existants afin de démontrer des améliorations principalement par rapport à leur temps d'exécution, à l'indistinguabilité des clefs produites et par le fait que certains de ces nouveaux algorithmes sont, à toutes fins pratiques, les meilleurs pouvant être qualifiés d'asymétriques. Notre théorème sur l'exactitude d'un de nos meilleurs générateurs de clés à porte dérobée est une généralisation d'un théorème de Boneh, Durfee et Frankel sur l'information partielle de l'exposant de déchiffrage.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTATION

Formatting of theorems and other claims or examples

- $\square$ ends an example.
- $\blacksquare$ ends a proof.

Bit strings

- Let ϵ be a variable which length is at least t bits. Then $\epsilon\rceil^t$ denotes the t most significant bits (MSB) of ϵ and $\epsilon\rfloor_t$, its t least significant (LSB) ones.
- Let r and s be strings of bits. Then $r\!:\!s$ denotes the bit string formed by the concatenation of the first two.

Sampling

- Let $x \in_R S$ denote the random sampling of variable x, where the probability distribution function is defined alongside with the set S.
- Let $x \in_U S$ denote the uniformly random sampling of variable x in the set S.
- Let $[A, ..., B] \subset \mathbb{Z}$ be the range of integers from A to B.
- A random prime p is denoted as **rp** p.

Divisibility

- Denote as $a|b$ that a divides b.
- Denote as $a^k||b$ that a^k divides b exactly, i.e. $a^k|b$, but $a^{k+1} \nmid b$.

Referring to (in)equations

- LHS refers to the *left hand side* of an (in)equation
- RHS refers to the *right hand side* of an (in)equation

Functions

- Let f and g be functions such that the image of g and the domain of f are the same. Then $f \circ g$ denotes function composition: $f \circ g : dom(g) \rightarrow ima(f)$.
- $\lg(n)$ denotes the base 2 logarithm of the variable n.
- $\phi(n)$ is the Euler totient function: for $n = pq$, a product of two prime numbers, $\phi(n) = (p-1)(q-1)$.
- Let $\gcd(x, y)$ be the greatest common divisor of the integers x and y.
- Let $\operatorname{lcm}(x, y)$ be the least common multiple of the integers x and y.

Groups

- Let $ord(g)$ denote the order of a group element g, as detailed in Theorem 2.2.2.

Additional notation for RSA backdoors

- Let $n = pq$ be an RSA modulus where $|p| = |q| = k$ and $q < p < 2q$.
- Let (n, ϵ) and (n, e) be RSA public keys with corresponding private keys, δ and d.
- Let κ be such that $\epsilon\delta - \kappa\phi(n) = 1$.

CONTRIBUTIONS OF AUTHORS

Signature padding

Chapter 3 shows how to construct a practical secure signature padding algorithm for arbitrarily long messages from a secure signature padding algorithm for fixed-length messages. This new construction is based on a one-way compression function respecting the "division intractability assumption" and is a direct improvement of the algorithm of Coron, Koeune and Naccache [CKN00].

By practical, it is meant that the new algorithm can be instantiated using dedicated compression functions and without chaining. This algorithm also allows precomputations on partially received messages. Finally, an instantiation of our algorithm is given using SHA-1 and PKCS #1 ver. 1.5.

Contributions. This padding algorithm comes from a collaboration with Jean-Marc Robert [AR01]. The author of this work is the main contributor of this paper.

Backdoors in key generation

As there has been various results on backdoors or related topics for the past 20 years, one of the goals of this work is to unify the field of study of backdoors. Chapter 4 initiates accomplishing this goal by establishing a definition of backdoors. Notions of indistinguishability of dishonest keys with respect to honest keys are abstracted.

Definitions of measures of quality for backdoors embedded in public keys are given. The measures that were found relevant for indistinguishability are in terms of number, distribution, and actions which the distinguisher (adversary or legitimate user) may take. These include key regeneration, reverse-engineering, and side channel analyses, as for timing and resource usage. Justification and examples are provided for each measure. Chapter 5 develops measures relevant to assessing the quality of backdoors.

Furthermore, new backdoors are developed such that they supersede other existing ones, with respect to the measures provided. To obtain such a superseding RSA backdoored key generator, a new extension of Wiener's Theorem (which originates from earlier works) is used, as well as a hereby contributed generalization of a theorem of Boneh, Durfee and Frankel, itself based on Coppersmith's lattice factorization. In parallel, this work presents how the same principles can be applied to obtaining asymmetric backdoors as well as backdoors for ElGamal key generation. Chapter 7 presents and gives the analysis of the algorithms that we developed.

Contributions. The formalization and uniformization of the definitions pertaining to backdoors is mainly due to the author of this work. This originated from discussions with the collaborators that follow. The extension of Wiener's Theorem originates from earlier work from Claude Crépeau and Alain Slakmon and was finalized in collaboration with the author of this work. It is also the case for most of the symmetric backdoors for RSA that are not attributed to other authors. The strongest symmetric backdoor for RSA is mainly due to the author of this work. This work has been submitted for publication [ACS].

The symmetric backdoor for ElGamal originates from earlier work from Claude Crépeau and Raza Ali Kazmi and was finalized in collaboration with the author of this work, and is based on the generalization of the theorem of Boneh, Durfee and Frankel. This generalization is a contribution of the author of this work. The asymmetric backdoors for RSA and ElGamal come from a collaboration of the author of this work with Raza Ali Kazmi and Claude Crépeau [ACK].

Furthermore, the author brought together the various definitions and corresponding algorithms of the field of the theoretical study of backdoors [Arb]. Besides the results of Crépeau et al., major contributions are due to Young and Yung [YY96, YY97a, YY97b, YY05a, YY05b]. Chapter 6 presents and gives the analysis of the existing algorithms, which have theoretical interest. The last section of Chapter 7 summarizes and compares the various algorithms.

CHAPTER 1
Introduction

The goal of this thesis is to broaden and compile knowledge on two mathematical aspects of the security of the rsa cryptosystem. The first of these two aspects is the well-established field of padding schemes, which is the topic of Chapter 3. The other aspect is the generation of public keys that are embedded with a backdoor. This is the topic of the remaining chapters, that is, Chapters 4 to 7.

Before further details, it is useful to understand the context of these results. First, the history of public-key cryptosystems is briefly recalled. A more detailed descriptions of the RSA cryptosystem and of the specific problems at hand follow.

1.1 Brief history of public-key cryptosystems

The invention of asymmetric (or public-key) algorithms is credited to Martin Hellman and two of his students of the time, Ralph Merkle and Whitfield Diffie. The earliest publicly available asymmetric construction are Merkle's Puzzles which were theoretically defined in 1974 [Mer78]. This construction is an answer to the problem of reestablishing a compromised secure connection between two computers.

Merkle's Puzzles are as follows. As usual in cryptology, let Alice and Bob be legitimate parties wishing to communicate, and let the eavesdropper be the malicious party wanting to compromise this communication. Bob sends a large number of puzzles to Alice. The solution of each puzzle consists of an identifier and a secret session key encrypted with a weak algorithm. Alice chooses a random puzzle and decrypts with a brute force algorithm. Then, she sends the identifier back to Bob, who learns which session key is to be used. An eavesdropper's best strategy is to decrypt all the puzzles until the right one is found, thus an eavesdropper's workload is considerably larger than Alice's. Figure 1–1 illustrates how Merkle's Puzzles relate to public-key cryptosystems by solving the said compromised secure connection problem.

$$\begin{tikzcd}
\textbf{Alice} & \textbf{Bob} \arrow[d] \\
\text{Problems 1 to } n \arrow[d, "\text{selects one randomly}"'] & \begin{array}{c}\text{Problems 1 to } n \\ \text{each with } (ID, key)\end{array} \arrow[l] \arrow[dd, "\text{knows the construction}"] \\
\text{Problem } i \arrow[d, "\text{solution by brute force}"'] & \\
(ID, key) \arrow[uur, "ID"'] & key
\end{tikzcd}$$

Figure 1–1: Merkle's Puzzles. After Alice transmits ID, both Alice and Bob have the key. An eavesdropper may learn Problems 1 to n and ID, but, except with considerable difficulty, not key.

Since the original one, many other theoretical solutions have appeared. Table 1–1 represents a non-exhaustive sample of public-key cryptosystems. Common types of cryptosystems are key agreement, encryption and signature algorithms. Key agreement allows more than one party to agree on a common key such than its value is influenced by all legitimate parties but no other. Encryption allows to make information unreadable without the possession of special knowledge, in the form of a key. Digital signatures ensure that only the claimed signer of an information can be identified as such.

construction	type	security	indirect attacks
[Coc73] (classified)	encryption	n^{th} root modulo n computation [1]	
[Mer78], 1974	key agreement	many puzzles	
[DH76]	key agreement	discrete logarithm	
[RSA78]	encryption, signature	e^{th} root modulo n computation [2]	chosen-ciphertext, weak keys
[MH78]	encryption, signature	knapsack	trapdoor discovered [Sha82] (broken) [3]
[Rab79]	encryption	integer factorization	
[GM84]	encryption	quadratic residuosity	
[ElG85]	encryption, signature	discrete logarithm	chosen-ciphertext
[BG85]	encryption	integer factorization	chosen-ciphertext
[Mil85], [Kob87]	key agreement, encryption, signature	discrete logarithm on elliptic curves	
[BR96]	signature	RSA78, Rab79	chosen-ciphertext
[CCKL01]	key agreement, encryption	DH-DP or MSCSP on braid group [4]	
[RS03]	key agreement, encryption, signature	discrete logarithm on torus group	
[RS06]	encryption	isogenies	

Table 1–1: Theoretical overview of public-key cryptography.

[1] Very similar to RSA, but with non-equivalent encryption and decryption functions. It was classified by the GCHQ.

[2] If integer factorization is feasible, RSA is not secure. The converse is unknown: does the computation of the e^{th} root modulo n allow the factorization of n?

[3] Shamir showed that from the public-key, a *trapdoor* can be computed, so that any plaintext can be easily computed from the corresponding ciphertext.

[4] DH-DP stands for Diffie-Hellman type Decomposition Problem and MSCSP, for Multiple Simultaneous Conjugacy Search Problem.

In the table, the *security* column gives the problem on which conjectured average-case hardness the cryptosystem's security rests. This is also called the *security assumption.* A *direct attack* then consists in solving the hard problem on which the security is based. The *indirect attacks* column gives a (non-exhaustive) list of attacks to which the cryptosystem is vulnerable, aside from the direct attack. This column reflects whether there has been significant theoretical research in finding indirect ways of breaking a construction [5] . Indirect attacks usually include semantic security (ciphertext indistinguishability), side channel analysis, weak keys, and man-in-the-middle, as well as attacks on auxiliary cryptographic protocols, such as standard hash functions and padding algorithms.

Overall, public-key cryptography was independently co-invented in the early 1970's. In academia, it was created by Merkle, Diffie and Hellman. As disclosed in 1997, in the British intelligence agency "Government Communications Headquarters" (GCHQ), it was created by Ellis, Cocks and Williamson [Way97].

Note on probabilistic encryption and randomness. In cryptography, the intuitive aim is to scramble messages in order to conceal their contents from unintended readers. In this context, randomness has been used to a large extent. In 1984, Shafi Goldwasser and Silvio Micali introduced the first probabilistic cryptosystem [GM84]. A plaintext is encrypted to one of many possible ciphertexts in such a way that it is as difficult as a (conjectured) hard problem to obtain any information about the plaintext. Although the original RSA is not probabilistic, randomness can

[5] *Quand Dieu ferme une porte, il ouvre une fenêtre.* (proverb)

be introduced via padding schemes, as detailed in Subsection 1.2.2 for signatures and at the end of Subsection 2.2.8 for encryption. Another use of randomness is the generation of random keys. Such keys may be symmetric (e.g. DES) or asymmetric (e.g. RSA).

1.2 The RSA cryptosystem

Table 1–1's fourth row lists the RSA cryptosystem, named after the initials of its inventors Ron Rivest, Adi Shamir and Leonard Adleman, [RSA78]. This cryptosystem is historically recognized as the first public-key encryption and signature algorithm.

It is on this cryptosystem that the rest of this work focuses. Chapter 2 gives mathematical background. Chapter 3 elaborates on *signature padding schemes*. Chapters 4 to 7 concerns the generation of keys that contain a hidden *backdoor*.

1.2.1 Definition

Informally, a trapdoor function is a function that is easy to compute in one direction, but hard to compute in the other one (the inverse function), unless a special information (a key) is known, in which case it is easy to invert. The RSA function is a well-known candidate trapdoor function. Because the RSA function can be employed either for encryption or digital signature, denote it in general by e_K, for some key K (the key, K, is the trapdoor key). Not that the knowledge of K is not necessary to use the encryption function, e_K. Its corresponding inverse is denoted by d_K. For all $x \in \mathbb{Z}_n$, it holds that

$$d_K \circ e_K(x) = x \text{ and } e_K \circ d_K(x) = x$$

as shown in Subsection 2.2.4.

Definition 1.2.1 *The* RSA **cryptosystem** *is a family of (candidate) trapdoor permutations. It is specified by:*

- *The* RSA *key generator Gen, which on input 1^k, randomly selects 2 distinct k-bit primes p and q and computes the modulus $n = p \cdot q$. It randomly picks an encryption (or signature verification) exponent $e \in \mathbb{Z}^*_{\phi(n)}$ and computes the corresponding decryption (or signature) exponent d such that*

$$e \cdot d \equiv 1 \bmod \phi(n).$$

  The generator returns $Gen(1^k) = (n, e, d)$.

- *The encryption (or signature verification) function $e_K : \mathbb{Z}_n \to \mathbb{Z}_n$ defined by*

$$e_K(x) = x^e \bmod n$$

  and more specifically, either $Encr(x) = e_K(x)$ or $Verif(x, y) = [e_K(x) == y]$, where "==" denotes Boolean equality. In practice, $Verif$ has an additional operation: the comparison of the message computed from the signature, x, with the message, y, which is transmitted with x, as defined below.

- *The decryption (or signature) function $d_K : \mathbb{Z}_n \to \mathbb{Z}_n$ defined by*

$$d_K(y) = y^d \bmod n$$

  and more specifically, either $Decr(y) = d_K(y)$ or $Sign(y) = (d_K(y), y)$.

Therefore, $K = (p, q, d, e)$ can be used as trapdoor key.

The RSA cryptosystem is used for digital signatures or encryption. In both cases, (n, e) is the public key and d is the private key. The sender of a message uses the recipient's public key e to encrypt the message; the corresponding decryption exponent d decrypts it. The sender of a message uses its private key d to produce a signature sent with the message; the public exponent e verifies that they correspond. For RSA, the trapdoor key therefore consists of both public and private keys.

Direct attack on RSA

The security of RSA is based on the conjectured average-case hardness of computing the e^{th} root modulo n. This relates to the hardness of the factorization of integers, of the form given in Definition 1.2.1. May recently showed that computing the RSA private key is (deterministically) equivalent to factoring [May04]. The probabilistic equivalence was already known in the original RSA paper.

Concretely, a direct attack on RSA consists in computing an e^{th} root modulo n given (n, e). However, it is only known that if integer factorization is feasible, then RSA is not secure; the converse is unknown. It may be the case that it is not necessary to find d to inverse the exponentiation by e. In other words, taking the e^{th} root may be easier than finding d, i.e. than factoring.

1.2.2 RSA signature padding

When RSA is used for signing, it is common practice to use the hash-and-sign paradigm: the information is hashed down before it is signed. This is because it is more efficient to exponentiate a shorter, hashed, number. The security of the signature algorithm is maintained given that the hash function, μ, is *collision-free.*

The hashed message is however subject to an additional step: it is padded via an encoding function, *enc*, in order to use the full length of the RSA modular exponentiation. Besides, padding is also used in a similar way for encryption. Therefore, a particular class of attack on RSA targets the security of the padding algorithm. Overall, the security of the signature algorithm is maintained given that μ is collision-free and given the security of the encoding function, *enc*.

Relation to Definition 1.2.1

The RSA signing function can be conceived as follows, from a theoretical (abstract) point of view to a progressively more applied (implementable) one. At the most abstract level given by Definition 1.2.1,

$$Sign(y) = (y^d \bmod n, y).$$

In the next level of implementation, one realizes that some messages are larger than n, so their length has to be reduced via a hash function, μ. This hash function is useful for cryptographic signing only if it is collision-free: it is difficult to find two of its inputs that hash to the same output. Otherwise, a signature for a given message could be (most likely illegitimately) reused for another one, which is an example of signature *forgery*. This more implementable signing RSA function is

$$Sign(y) = (\mu(y)^d \bmod n, y).$$

A final problem is patched via encoding. The output length of standard functions, μ, may not be exactly $|n| = 2k$ (typically, $|\mu(y)| < 2k$). To use the entire modular exponentiation length, an encoding function (also known as padding function) is used. This more applicable signing RSA function is

$$Sign(y) = (enc \circ \mu(y)^d \bmod n, y).$$

Formal definition of signature padding

More formally, let μ be a randomized compression function taking as input a message of size t, using r random bits, and outputting a message digest of length l:

$$\mu : \{0,1\}^r \times \{0,1\}^t \rightarrow \{0,1\}^l$$

where μ is useful if $l \leq 2k < t$, so that it hashes down the length of messages to at most the length of n, and if it is collision-free, i.e. it is computationally infeasible to find two inputs on which μ produces the same output. Also, the random bits, r, are useful to counter an attack that will be explained later.

Let *enc* be an encoding function taking as input a message digest of size l, and outputting an encoded message of length $2k$:

$$enc : \{0,1\}^l \rightarrow \{0,1\}^{2k}$$

where *enc* is useful in cases where $l < 2k$.

Overall, a padding algorithm for RSA computes the following function.

$$enc \circ \mu : \{0,1\}^r \times \{0,1\}^t \rightarrow \{0,1\}^{2k}$$

Attacks on padding schemes

Attacks on padding algorithms include chosen-ciphertext attacks (CCA), as shown in Figure 1–3. They are a special case of a man-in-the-middle (MITM) attack, as shown in Figure 1–2, where the adversary has access to a signature oracle (sometimes referred to, by abuse of language, as a decryption oracle). In practice, CCA can be deployed as a *lunch time* attack, where the adversary has access to a signing device at a time where its legitimate user leaves it unattended. A concrete example is the one of a smart card which may be under the full control of the adversary, for a limited amount of time.

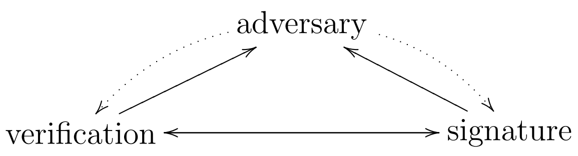


Figure 1–2: Man-in-the-middle attack: the adversary can observe and intercept messages. In the more general setup (that includes the dotted lines), it can also modify the messages.

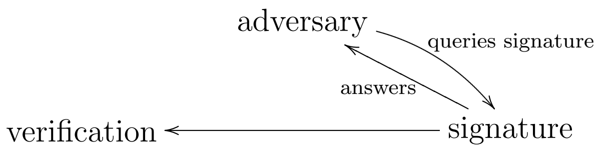


Figure 1–3: Chosen-ciphertext attacks (CCA) on a signature algorithm: the signing algorithm acts as a signature oracle for the adversary. Moreover, if the queries are not chosen all at once, the figure can be seen as showing an *adaptive* version of this attack, where the adversary can adapt its strategy w.r.t. the oracle's previous answers.

Contribution

Chapter 3 consists of an improvement on a signature padding algorithm of Coron, Koeune and Naccache [CKN00].

1.2.3 Backdoors in public key generation

Consider a special type of indirect attack, called a weak key attack. In general, a weak key is one that makes the cryptosystem behave in an undesirable way: weak keys constitute a weakness of the cryptosystem. A particular type of weak key attack is one in which the designer produces keys that contain a hidden backdoor.

Such a backdoor attack is as follows. In many cases, the end users get a piece of hardware or software which generates the public and private key pairs. These key generators may have been developed by malicious designers. In such cases, the designer may have implemented the generators in such a way that the designer can retrieve the private key corresponding to any generated public key. Such generators are said to be *backdoored* and to generate *weak hidden keys*.

Contribution

Chapter 4 provides a more complete and uniform way of defining backdoored key generators and complementary notions. Chapter 5 develops a set of measures that allow to analyze such algorithms in a uniformed manner. Chapter 6 repertories the existing theoretical algorithms, with respect to these measures. Chapter 7 improves on existing algorithms and compares all the presented algorithms.

The Doge of Venice vs. Casanova

Consider the following scenario. The Doge of Venice is in need of evidence in order to throw Casanova out of the City. The Doge has partial control over the City's system of secret message exchange (essential for commerce). This is a rudimentary public-key system where a first party sets up a public mailbox with an open padlock, to which this first party owns the key. Once such a mailbox is set up, any other party can securely send a message to the owner of the mailbox. Casanova uses the system to receive messages from lovers (or associates) with whom he cannot have a private word (no meeting could be arranged to exchange secret information).

The control of the Doge over the system is in its *magical* key manufacturing, where a key is understood to include the key itself and the corresponding padlock on a box. The magician-keysmith produces padlocks that react to a magic spell, known to the Doge only. When this spell is applied to a padlock, a copy of the corresponding key appears out of thin air [6] . Officially, the keysmith is thought to be honest and appears to produce locks and keys that behave normally. Such a padlock on a box therefore is part of a *hidden weak key*. The Doge's spell uses the weakness of the City's padlocks as a *backdoor* to retrieve their keys.

The Doge's attack is as follows. Casanova's padlock reacts as any other to the spell: the Doge retrieves Casanova's key. The Doge is only left with collecting enough

[6] "Any sufficiently advanced technology is indistinguishable from magic." (A.C. Clarke, *Profiles of The Future*, 1961.)

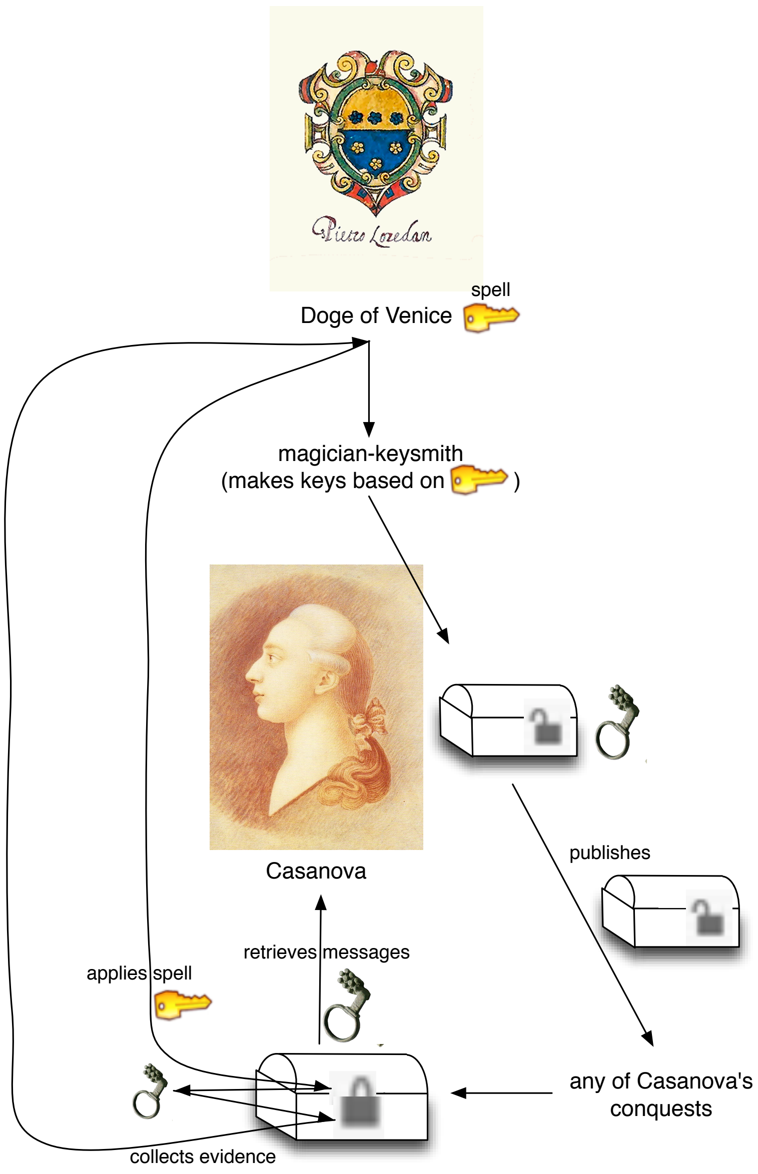


Figure 1–4: The Doge of Venice's scheme to cast out Casanova.

evidence of Casanova's behaviour in order to cast him out. For an illustration, refer to Figure 1–4.

1.3 Chapter notes

Brief history. The information on Merkle's Puzzles is taken from [Mer78]. The discoveries of the GCHQ are commented in [Way97] and the title of the original 1973 secret report is *A Note on Non-Secret Encryption* [Coc73].

Theoretical overview of public-key cryptography. The information on public-key algorithms in Table 1–1 is taken from the RSA Labs Crypto FAQ [RSAa] as well as directly from the original papers, as cited.

CHAPTER 2
Background on RSA

2.1 History

As introduced in Section 1.2, RSA, for their inventors Rivest, Shamir and Adleman, [RSA78], is a public-key cryptographic algorithm. Historically, it is recognized as the second such algorithm, although believed to be the first for nearly 20 years.

In earlier cryptosystems, the encryption rule, e_K, and decryption rule, d_K, were either the same or were easily deduced from one another. If one of the rules is exposed, then security collapses. These cryptosystems are thus called *symmetric*. Another drawback of symmetric cryptography is *key exchange*. The key, K, used by the encryption and decryption rules must be exchanged secretly, through a secure channel, in order for the cryptosystem to become usable. A secure channel may be expensive or unavailable if the parties who wish to communicate are far apart.

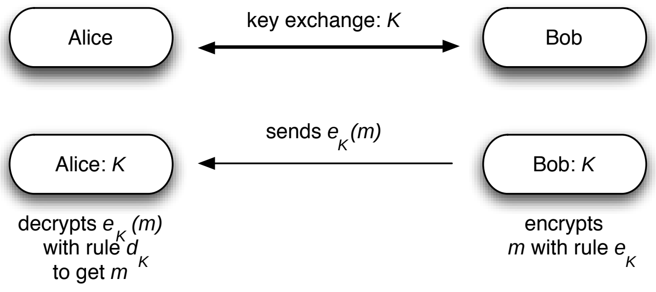
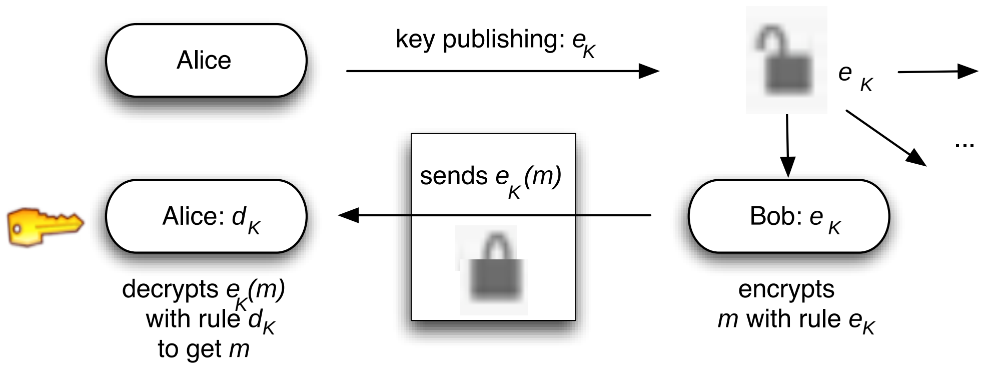

Alice
key exchange: K
Bob
sends $e_K(m)$
Alice: K
Bob: K
decrypts $e_K(m)$ with rule d_K to get m
encrypts m with rule e_K

Figure 2–1: Symmetric cryptography. Top, key exchange via a secure channel. Bottom, encryption with exchanged key, K, transmission of encrypted message via a public channel, and decryption with the same K.

In *asymmetric cryptography*, the encryption rule can be revealed while leaving the decryption rule computationally infeasible to deduce. As such, it was originally called *non-secret encryption* This eliminates the need for key exchange via a secure channel, and the decrypting party (Alice, in Figure 2–1) can publish the encryption rule, e_K, in a public directory, for any other party's usage. Thus, this is also called *public-key cryptography.* It is as if Alice publicly provides an open padlock. Bob can lock a box with it, but only Alice can open it.



Alice
key publishing: e_K
e_K
...
sends $e_K(m)$
Alice: d_K
Bob: e_K
decrypts $e_K(m)$ with rule d_K to get m
encrypts m with rule e_K

Figure 2–2: Asymmetric cryptography. Top, Alice transmits the public encryption rule, e_K, to a public directory, where Bob, amongst others, can obtain e_K. Bottom, Bob encrypts with e_K, transmits the encrypted message via a public channel, and Alice decrypts with the decryption rule, d_K.

2.2 Theory

2.2.1 Computational security

The security of asymmetric cryptosystem is always *conditional.* Their *computational security* is studied, as to consider a computationally bounded adversary.

Suppose that a ciphertext y is observed by an adversarial eavesdropper who, as anyone, has access to e_K. With unbounded computational resources, the adversary can encrypt every possible plaintext m until the one m such that $y = e_k(m)$ is found.

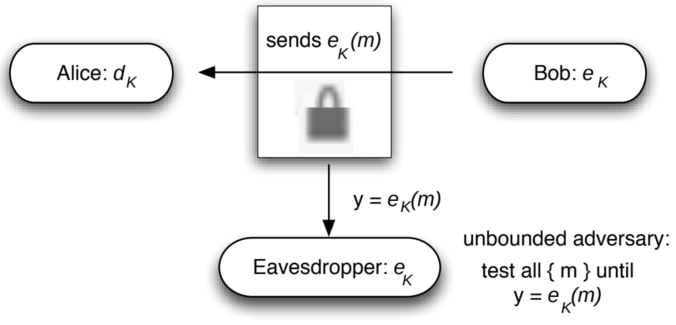


Figure 2–3: The computationally unbounded eavesdropper's attack on an asymmetric cryptosystem.

2.2.2 Trapdoor one-way function

First, an asymmetric cryptosystem needs to be such that the private d_K is computationally infeasible to compute from the public e_K. This will be developed in Subsection 2.2.7.

More generally, Subsection 2.2.1 shows that an asymmetric cryptosystem is required to be such that the original encrypted message, m, is computationally infeasible to compute from the encrypted message, $y = e_k(m)$. This holds for the eavesdropper and, more generally, for anyone other than Alice. Moreover, it is also needed that $y = e_k(m)$ is easy to compute. This holds for Bob.

That $y = e_K(m)$ is easy to compute while $m = e_K^{-1}(y)$ is infeasible corresponds to the definition of a *one-way function*: e_K is easy to compute and hard to invert. Furthermore, that $m = d_K(y)$ is easy to compute corresponds to the definition of a *trapdoor one-way function*: e_K is easy to compute and hard to invert unless a *trapdoor*, K, is known. Knowing the trapdoor provides Alice with the decryption function, d_K.

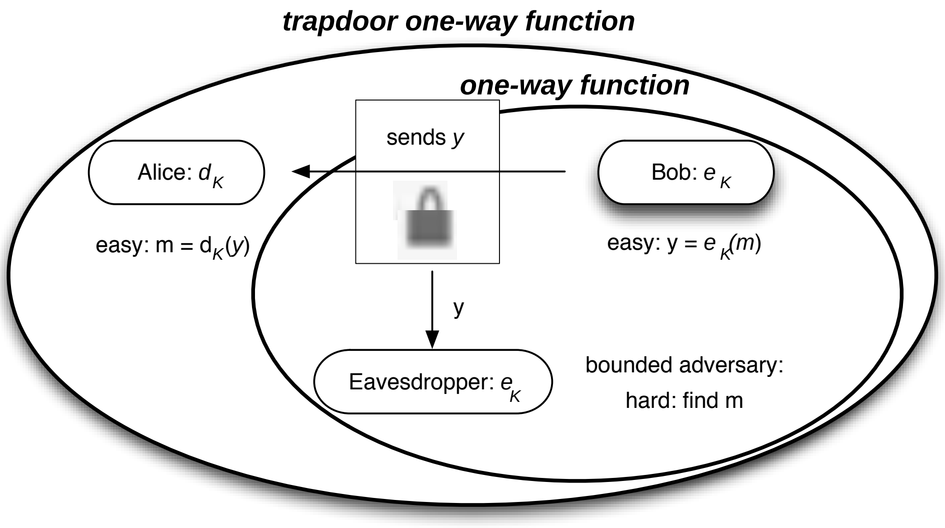


Figure 2–4: Computational bounds on parties involved in an asymmetric cryptosystem. Encryption is easy for Bob. Decryption is only easy for Alice, assuming the eavesdropper is computationally bounded. From the point of view of only Bob and the eavesdropper, e_K is hence a one-way function. From Alice's point of view, e_K is a trapdoor one-way function.

To insure security, a symmetric key is randomly chosen in a keyspace. Similarly, a trapdoor function, e_K, is randomly chosen in a family of such keyed functions, $\mathcal{F}$. Then e_K is published and d_K, kept private.

2.2.3 Toward the definition of RSA

The encryption function of the RSA cryptosystem is an example of trapdoor one-way function. Let n be the product of two large primes, p and q. Define the function $e_K : \mathbb{Z}_n \to \mathbb{Z}_n$ to be $e_K(m) \equiv m^e \bmod n$, for $\gcd(e, \phi(n)) = 1$, where gcd is the greatest common divisor and $\phi(n)$ is the Euler totient function, which gives the number of positive integers less than n which are co-prime with n. The trapdoor is an efficient way of computing d (or is d itself) such that the inverse function, d_K, is of the form $d_K(y) = y^d \bmod n$, which computes $\sqrt[e]{y} \bmod n$.

2.2.4 The definition of RSA

The RSA cryptosystem and digital signature algorithms [RSA78], are based on the generation of two random primes, p and q, of roughly equal size, k bits, and the generation of random exponents, d and e. It holds that $de \equiv 1 \pmod{\phi(n)}$, where the RSA modulus, n, is the product $n = pq$. The pair, (n, e), is made public. This is illustrated in Figure 2–5. The EG key generator is given in Figure 7–11

Algorithm G_0 [Standard RSA key generation]

```
1: Pick random primes $p, q$ of appropriate size $= k$, and set $n = pq$.
2: repeat
3:     Pick a random odd $e$ such that $e < \phi(n)$.
4: until $\gcd(e, \phi(n)) = 1$.
5: Compute $d \equiv e^{-1} \bmod \phi(n)$.
6: return $K = (p, q, d, e)$.
```

Figure 2–5: Standard (honest) RSA key generation.

Algorithm e_K [RSA encryption of message m]

```
1: $e_K(m) \equiv m^e \bmod n$.
2: return $e_K(m)$.
```

Algorithm d_K [RSA decryption of message m]

```
1: $d_K(m) \equiv m^d \bmod n$.
2: return $d_K(m)$.
```

Figure 2–6: RSA encryption and decryption functions.

The encryption and decryption operations are given in Figure 2–6. To show that encryption and decryption are inverse operations, recall that $d \equiv e^{-1} \bmod \phi(n)$. Therefore

$$de = t\phi(n) + 1,$$

for some integer $t \geq 1$. Suppose that $m \in \mathbb{Z}_n^*$, then

$$\begin{aligned}
(m^e)^d &\equiv m^{t\phi(n)+1} \bmod n \\
&\equiv m^{t\phi(n)} m \bmod n \\
&\equiv 1^t m \bmod n \qquad (2.1) \\
&\equiv m \bmod n
\end{aligned}$$

where all the steps are direct, except for (2.1). It comes from Theorem 2.2.1: if $m \in \mathbb{Z}_n^*$, then $m^{p-1} \equiv 1 \bmod p$ so $m^{\phi(n)} \equiv 1 \bmod p$ and the same holds modulo q. Then, by Theorem 2.2.3, $m^{\phi(n)} \equiv 1 \bmod n$.

Theorem 2.2.1 (Fermat's Little Theorem) *Suppose p is prime and $m \in \mathbb{Z}_p$. Then $m^p \equiv m \bmod p$.*

The following theorem bears Lagrange's name though only the then future development of group theory allowed to prove its general form, as documented in [Rot01]. Although it predates it, Fermat's Little Theorem can be seen as its corollary. This is because $\mathbb{Z}_n^*$ is a multiplicative group of order $\phi(n)$.

Theorem 2.2.2 (Lagrange's Theorem) *Let the order of a group be the number of its elements, and the order of an element m of a multiplicative group be the smallest positive integer r, denoted $ord(m)$, such that $m^r = 1$.*

Suppose that $g \in G$, a multiplicative group of order n. Then $ord(g)|n$.

Theorem 2.2.3 (Chinese Remainder Theorem for $n = pq$, with $p \neq q$) *The simultaneous congruences $m^x \equiv a \bmod p$ and $m^x \equiv b \bmod q$ have a unique solution modulo n:*

$$m^x \equiv a \cdot q \cdot [q^{-1} \bmod p] + b \cdot p \cdot [p^{-1} \bmod q] \bmod n.$$

The remaining cases are $m \in \mathbb{Z} \setminus \mathbb{Z}_n^*$, for which the preceding theorem is also useful. Without loss of generality, suppose that m is a multiple of p, so $m = \ell p$ and $\ell < q$ since $m < n$. Then $m^{t\phi(n)+1} \equiv 0 \bmod p$ so the preceding $a \equiv 0$. Also, $m^{t\phi(n)+1} \equiv m \bmod q$, so $b \equiv m \bmod q$.

$$\begin{aligned} m^{t\phi(n)+1} &\equiv 0 + [m \bmod q] \cdot p \cdot [p^{-1} \bmod q] \bmod n \\ &\equiv [\ell \bmod q] \cdot [p \bmod q] \cdot p \cdot [p^{-1} \bmod q] \bmod n \\ &\equiv \ell \cdot p \bmod n \equiv m \end{aligned}$$

Computing d with the trapdoor. The decryption exponent can be computed efficiently, as part of the key generation. The encryption function is

$$e_K(m) = m^e \bmod n,$$

and the decryption one is

$$d_K(y) = y^d \bmod n.$$

Therefore, the *trapdoor* (as defined in Subsection 2.2.2) to the one-way function e_K is the factorization of $n = pq$, or equivalently $\phi(n)$. With this trapdoor and e, the RSA key generator computes $d \equiv e^{-1} \bmod \phi(n)$ as follows.

For a positive integer a, the set $\mathbb{Z}_a$ can be interpreted as a ring, with modular addition and mutiplication. Let $a = \phi(n)$, then consider $\mathbb{Z}_{\phi(n)} = \{0, ..., \phi(n) - 1\}$ as a ring. The inverse of $e \in \mathbb{Z}_{\phi(n)}$ exists if and only if $\gcd(e, \phi(n)) = 1$, otherwise e is a zero divisor. The Euclidian division Algorithm computes the gcd of two integers, so it can be used by the generator to test the condition, $\gcd(e, \phi(n)) = 1$. The extended Euclidian division Algorithm also computes the gcd, as well as two integers, d and κ, such that $ed + \phi(n)\kappa = \gcd(e, \phi(n))$. Thus, if the condition holds, it finds $d \equiv e^{-1} \bmod \phi(n)$.

Computing the encryption and decryption functions. Given the appropriate keys, i.e. the exponents e and d, the extent of the computations involved by the encryption and decryption functions consists in modular exponentiations. The straightforward exponentiations followed by a modulo take times $\mathcal{O}(e + T(n))$ and $\mathcal{O}(d + T(n))$, where $T(n)$ is the complexity of multiplication modulo n.

However, a modular exponentiation by e is feasible in time $\mathcal{O}(\lg e \cdot T(n))$. This can be achieved using that for any integers, a and b,

$$a \cdot b \bmod n \equiv ((a \bmod n) \cdot (b \bmod n)) \bmod n$$

and that an exponent has a binary notation form. This is the method known as *exponentiation by squaring* or *square and multiply exponentiation.* For this, consider the n-bit e in binary notation:

$$e = \sum_{i=0}^{n-1} a_i 2^i$$

where $a_i \in \{0,1\}$ and $a_{n-1} = 1$ (the latter by the standard convention on bit length). Then

$$\begin{aligned} m^e &= m^{\left(\sum_{i=0}^{n-1} a_i 2^i\right)} \\ &= \prod_{i=0}^{n-1} \left(m^{2^i}\right)^{a_i}, \end{aligned}$$

therefore

$$\begin{aligned} e_K(m) &= m^e \bmod n \\ &\equiv \prod_{i=0}^{n-1} \left[\left(m^{2^i}\right)^{a_i} \bmod n\right] \bmod n. \end{aligned}$$

Clearly, d_K is computable similarly.

Lexicon: vocabulary and variables

As RSA can be used in the two contexts of encryption and signature, the names given to the exponents are dependent on these contexts.

private key: denoted d, is the decryption key in the context of an encryption algorithm and the "signing key" in the one of a signature algorithm

public key: denoted e, is the encryption key in the context of an encryption algorithm and the "verification key" in the one of a signature algorithm

Additional notation for backdoors. Let $n = pq$ be an RSA modulus where $|p| = |q| = k$ and $q < p < 2q$ [1] . Let (n, ϵ) and (n, e) be RSA public keys with corresponding private keys, δ and d. Let κ be such that $\epsilon\delta - \kappa\phi(n) = 1$.

[1] In practice, the bounds of Table 2–1 (Section 2.3) are also applied.

2.2.5 Efficiency of key generation

Parts of Chapters 4 to 7 concern efficiency of altered key generators. For comparison, the corresponding properties for honest RSA key generation follow.

Theorem 2.2.4 (Prime number theorem) *Let $\pi(n)$ be defined as the number of prime numbers less than or equal to n. Then, as $n \to \infty$, it holds that*

$$\pi(n) \sim \frac{n}{\ln n}$$

in asymptotic (big O) notation, and, for $n \geq 17$ on the LHS and $n > 1$ on the RHS,

$$\frac{n}{\ln n} < \pi(n) < 1.25506\,\frac{n}{\ln n}.$$

Therefore, putting aside constants, Theorem 2.2.4's first approximation for $\pi(n)$ can be used for any practical, large enough, n. An example follows.

Example 2.2.5 For a 1024-bit RSA modulus $n = pq$, p and q are chosen as 512-bit primes. A 512-bit random integer is prime with probability approximately

$$\frac{1}{\ln 2^{512}} \approx \frac{1}{355}.$$

On average, given a set of 355 random 512-bit random integers, one of them is prime. This is a large enough fraction for the random generation of integer to be considered efficient in order to generate prime numbers. □

Example 2.2.6 The probability that a random k-bit integer is prime is

$$\frac{\pi(2^k)}{2^k} \approx \frac{1}{\ln(2^k)} = \frac{2}{2k\ln 2} \approx \frac{3}{2k} \approx 2^{-\lg k}$$

where the last approximate equality will, throughout Chapters 4 to 7, be the mainly used form. □

2.2.6 Number of generable keys

Parts of Chapters 4 to 7 concern the number of keys generable by altered key generators. For comparison, the corresponding properties for honest RSA key generation are given in this section.

Example 2.2.7 The number of k-bit primes is

$$\#\{p\} \quad \approx \quad \pi(2^k) \approx \frac{2^k}{\ln(2^k)} \approx 2^{k-\lg k}$$

where the first LHS approximation is due to $\#\{p\}$ referring to primes of exactly k bits, while $\pi(2^k)$ refers to all primes of length k bits or less. □

For instance, for $2k$-RSA primes, $\#\{p\} \approx \pi(2^k)$ is an approximation in two ways. Firstly, only primes of length exactly k are used. Secondly, standard precautions reject certain primes, as Section 2.3 details. In both cases, the total number is affected by a constant factor. Asymptotically, the approximate equality is not affected.

Example 2.2.8 The parameter d can be generated by randomly picking an integer in $\mathbb{Z}^*_{\phi(n)}$. The number of such integers is the number of integers co-prime to $\phi(n)$, therefore, it is $\phi(\phi(n))$. From [MvV01, Fact 2.102], it holds that

$$\begin{array}{rcl} \dfrac{n}{6\,\ln\ln n} < & \phi(n) & < n \\ \dfrac{n}{36\,(\ln\ln n)\,\ln\ln(n/(6\,\ln\ln n))} < & \phi(\phi(n)) & < n \end{array}$$

where $\ln \ln n$ is short hand for $\ln(\ln n)$, only used here in order to avoid confusion with parentheses.

Dividing n only by $\ln n$ does not affect the resulting order significantly. Therefore, $\phi(n)$ or $\phi(\phi(n))$ can be approximated by n in order to quantify the cardinality of a set or the complexity of an algorithm. Both these cases are similar: compared to the exponential size of n (in terms of $2k$), a division by $\ln n$ is not significant.

Moreover, the practical selection of p and q also provides a lower bound for both $\phi(n)$ and $\phi(\phi(n))$. In Table 2–1 (Section 2.3) states that p, q are chosen such that $p-1, q-1$ have large prime factors. Moreover, if the large factor of $p-1$ is r, then r is also chosen to have a large prime factor. This holds for both factors p and q. This allows the following bound is derived. Suppose that $|p-1| \approx |q-1| \approx k$, then $|\phi(n)| \approx 2k$. Similarly, if $|r-1| \approx k$, then $|\phi(\phi(n))| \approx 2k$. □

Example 2.2.9 From Examples 2.2.7 and 2.2.8, and because to each value of e corresponds exactly one inverse d, there are

$$\begin{aligned} \#\{(p,q,d,e)\} &= \#\{(p,q)\} \cdot \#\{d\} \\ &\approx \left(2^{k-\lg k}\right)^2 \cdot \phi(\phi(n)) \approx 2^{4k-2\lg k} \end{aligned}$$

honest keys. With $2k = 1024$, this resolves to approximately 2^{2030} honest keys. □

2.2.7 Provable security

Parts of Chapters 4 to 7 concern the retrieval of information hidden by altered key generators. This information relates to provable security features of RSA keys.

Known p, q implies known d, e. By design of the RSA cryptosystem, it is made so that knowing p, q implies knowing d, e. In the *Computing the trapdoor* of the preceding Subsection 2.2.4, it is shown how to compute d from e and $\phi(n)$. The knowledge of $\phi(n)$ equivalent to the one of the factorization, p, q, of n. Indeed by definition, it holds that $\phi(n) = (p-1)(q-1) = n - p - n/p + 1$ Taking the left and right hand sides, multiplied by p, and collecting the terms in p:

$$p^2 - (n - \phi(n) + 1)p + n = 0$$

has two solutions:

$$p, q = \frac{n - \phi(n) + 1 \pm \sqrt{(n - \phi(n) + 1)^2 - 4n}}{2}.$$

Known p, q is equivalent to known d, e. In the original RSA paper, a probabilistic equivalence between factoring n and the knowledge of d, e is given. If d, e is known then $ed - 1$ is a multiple of $\phi(n)$. Miller has shown that knowing a multiple of $\phi(n)$ allows to probabilistically factor n [Mil75].

May showed the corresponding deterministic equivalence: knowing a multiple of $\phi(n)$ allows to deterministically factor n [May04]. It is based on Coppersmith's algorithm [Cop97] for finding small roots of bivariate integer polynomials via lattice methods. Even so, in practice Miller's algorithm is used, because its expected time is likely to yield actual running times better than May's because lattice methods yield high time complexity.

Known bit of m implies known m. This work does not use the following theoretical results; they are cited here for the sake of the completeness of the listing of known security issues in RSA. From a theory point of view, predicting one bit of the encrypted message is as hard as breaking RSA [HN98]. This means that given $(n, e, m^e \bmod n)$, knowing one bit of m implies knowing the entire message m. This result's reduction is valid with an oracle yielding the physical i^{th} bit of m, as opposed to a more general oracle that would yield one bit of information on m.

This is an extension of a well-known result by Goldwasser, Micali and Tong. This earlier result states that the least significant bit of an encrypted message is as secure as the whole message [GMT82].

Breaking RSA may not be equivalent to factoring. A result of Boneh and Venkatesan provides evidence that breaking low exponent RSA may be easier than factoring [BV98]. Low exponent means small e, but the result applies as well for any smooth e, i.e. any e with prime factors smaller than a given bound. This result suggests a possible fundamental reason for the lack of progress in proving that breaking RSA is equivalent to factoring.

For instance, consider the case where $e = 3$ and $\gcd(\phi(n), 3) = 1$. Breaking RSA means computing cubic roots modulo n [2] The result in [BV98] establishes that if it can be shown that computing cubic roots modulo n is as hard as factoring n, then

[2] When $\gcd(\phi(n), 3) \neq 1$, it is well-known that computing cubic roots modulo n is as hard as factoring. Because $\gcd(\phi(n), 3) = 3$, it holds that $\phi(n) = 3k \equiv 0 \bmod \phi(n)$, for some $1 \leq k < \phi(n)$. Any integer $y < n$ can be written in the form $y = z^{ak+l}$ for $0 \leq a \leq 2$. Therefore $y^3 = z^{3ak+3l} = z^{3l}$, that is, for three different y, the cube is the same. An algorithm that outputs the cubic root modulo n of z^{3l} returns one of the three choices, so $x \not\equiv y \bmod n$ with probability $2/3$. If $x^3 \equiv y^3 \bmod n$, then knowing x and y yields a non-trivial factor $x - y$ of $x^3 - y^3 \equiv 0 \bmod n$ with probability $2/3$.

that proof will provide a factoring algorithm that does not make use of an oracle. Given these conditions, a cubic root (algebraic [3]) oracle cannot help factoring n.

2.2.8 Distribution properties of RSA

Parts of Chapters 4 to 7 concern the distribution properties of altered key generators. For comparison, the corresponding properties for honest RSA key generation are given in this section. Moreover, the RSA function itself is sometimes used to alter a key generator. Therefore, this section also serves as a reference for the properties of these alterations.

One inversion of a fixed e modulo $\phi(n)$. Some information on the distribution of the modular inversion of a fixed e was shown by Shparlinski [Shp04]. The formal version of the theorem is first given, to demonstrate the exactness of the interpretation that we will finally derive. Define the *discrepancy* D of a sequence of N points $(\gamma_j)_{j=1}^{N}$ of the unit interval $[0, 1]$ as

$$D = sup_{\mathcal{I}} \left| \frac{A(\mathcal{I})}{N} - |\mathcal{I}| \right|,$$

where the supremum is taken over the interval $\mathcal{I} = [\alpha, \beta] \subseteq [0, 1]$, $|\mathcal{I}| = \beta - \alpha$ and $A(\mathcal{I})$ is the number of points of this sequence which belong to $\mathcal{I}$. For a real α and an integer $k \geq 1$, let $\sigma_\alpha(k) = \sum_{d|k} d^\alpha$.

Theorem 2.2.10 [Shp04] *Let R and e be sufficiently large integers and let $\mathcal{R} \subseteq [R, 2R]$ be an arbitrary set of primes r such that* $\gcd(e, r - 1) = 1$. *Suppose that n*

[3] An algebraic oracle is such that it is restricted to arithmetic operations. For instance, it cannot return $x \oplus y$.

runs through all products $n = pq$ with $p, q \in \mathcal{R}$, $p \neq q$, and d is the RSA *exponent that is the inverse of e modulo $\phi(n)$. Then the discrepancy, $D(e, \mathcal{R})$, of the sequence $d/\phi(n)$ satisfies*

$$D(e,\mathcal{R}) \ll \begin{cases} \frac{e^{1/2}}{\#\mathcal{R}}\sigma_{-1/2}(e)\log e & \text{if } e \geq R, \\ \frac{R}{\#\mathcal{R}e^{1/2}}\sigma_{-1/2}(e)\log e & \text{if } e < R. \end{cases}$$

The cited paper provides the following interpretation. From the inequality $\sigma_{-1/2}(e) \leq \tau(e)$, where $\tau(e) \leq \log_2 e = e^{\frac{\log_2 \log_2 e}{\log_2 e}}$ is the number of positive integer divisors of e, it follows that $\sigma_{-1/2}(e) = e^{o(1)}$. Suppose that the admissible primes, $\mathcal{R}$, are dense: $\mathcal{R} \geq R/\log^A R$, for some constant $A > 0$. Then it also holds that

$$D(e,\mathcal{R}) \ll R^{-\varepsilon/2+o(1)}$$

for any fixed ε and any $e \in [R^{\varepsilon}, R^{2-\varepsilon}]$. This means that the distribution of the string formed by approximately the

$$0.5\varepsilon \log R \text{ MSB of } d$$

is exponentially close to the distribution of the string formed by the same number of most significant bits of randomly chosen integer in the interval $[0, \phi(n) - 1]$.

We reformulate this interpretation for the parameters that are useful in this work, as the following theorem. For a fixed and sufficiently large e and for a random choice of the RSA modulus $n = pq$, where p and q are k-bit admissible primes, the decryption exponent d, defined by $ed \equiv 1 \bmod \phi(n)$, is close to being uniformly distributed in the following sense. A fraction (linear in the length of n) of the most significant bits (MSB) of $e^{-1} \bmod \phi(n)$ is exponentially close to the distribution of the string formed by the same number of MSB of a randomly chosen integer in the interval $[0, \phi(n) - 1]$.

Theorem 2.2.11 ([Shp04], informal) *Suppose a fixed* $e \in [2^{4x(k-1)}, 2^{(2-4x)(k-1)}]$ *and sampled* $p, q \in_U [2^{k-1}, 2^k]$ *such that* $\gcd(e, p-1) = \gcd(e, q-1) = 1$*. Then for* x *sufficiently small, an approximate fraction of* x *of the* MSB *of* $e^{-1} \bmod \phi(n)$ *are distributed exponentially close to the distribution of the string formed by the same number of* MSB *of a randomly chosen integer in the interval* $[0, \phi(n) - 1]$.

A polynomial set of distributed e inversions modulo $\phi(n)$. Now suppose that e is limited to be of in certain set or of a certain size. Does a similar statement hold, for a given distribution on e and fixed n? Let us examine a series of cases that will build intuition and lead to the formulation of a computational assumption which is coherent with Theorem 2.2.11.

Remark 2.2.12 *Consider a given distribution of values of* e *in* $\mathbb{Z}^*_{\phi(n)}$*, for some fixed* n*. Next, consider the distribution formed by the set of their modular inverses. Then the elements of this set have the same distribution, up to relabeling, in* $\mathbb{Z}^*_{\phi(n)}$ *because the modular inversion function is a permutation.*

If the distribution is uniform, then the resulting distribution is uniform as well.

However, it is possible to construct distributions of e such that the inverses are easily distinguishable from the uniform distribution.

Example 2.2.13 Consider the set of all e, such that their inverses are small as to satisfy the conditions for δ in Wiener's Theorem (Theorem 6.3.1). Aside from being weak keys, these inverses are all correlated by their small sizes: such values of d are smaller than $n^{1/4}/3$.

$$(p, q, d_1, ..., d_{poly(k)}) \quad \text{where} \quad \gcd(d_i, \phi(n)) = 1 \text{ and } d_i < n^{1/4}/3 \tag{2.2}$$

$$(p, q, d'_1, ..., d'_{poly(k)}) \quad \text{where} \quad d'_i \in_U \mathbb{Z}^*_{\phi(n)} \tag{2.3}$$

where $poly(k)$ is a polynomial function in the size of n. Distribution 2.2 is efficiently **distinguishable** from the uniform distribution on $\mathbb{Z}^*_{\phi(n)}$ (Distribution 2.3). $\square$

In Distribution 2.2, the distinguisher knows (p, q), hence $\phi(n)$ so can check that $\gcd(d_i, \phi(n)) = 1$. Thus, it is capital that this condition be met in d_i's generation.

The "opposite" example equivalently yields a distinguishable distribution.

Example 2.2.14 Consider the set of all e that are themselves small as to satisfy the conditions for δ in Wiener's Theorem (Theorem 6.3.1): the modular inverses of the d_i are all smaller than $n^{1/4}$. Their inverses are correlated by Wiener's Theorem: a pair (n, d) yields the corresponding e, thus the factorization of n, which the distinguisher can check against (p, q).

$$(p, q, d_1, ..., d_{poly(k)}) \quad \text{where} \quad d_i \equiv e_i^{-1} \bmod \phi(n) \text{ and } e_i < n^{1/4}/3 \tag{2.4}$$

$$(p, q, d'_1, ..., d'_{poly(k)}) \quad \text{where} \quad d'_i \in_U \mathbb{Z}^*_{\phi(n)} \tag{2.5}$$

Thus (2.4) is efficiently **distinguishable** from the uniform distribution on $\mathbb{Z}^*_{\phi(n)}$. $\square$

Clearly, the distribution on e of Example 2.2.14 cannot be used directly to generate a distribution close to Distribution 2.5. Nevertheless, this can be fixed via a pseudo-random permutation.

Example 2.2.15 If d is then taken through a pseudo-random permutation, π_β, chosen randomly from a family of pseudo-random functions keyed with β, then

$$\begin{array}{lll} (p, q, \pi(d_1), ..., \pi(d_{poly(k)})) & \text{where} & d_i \equiv e_i^{-1} \bmod \phi(n),\ e_i < n^{1/4}/3 \\ & \text{and} & \gcd(\pi_\beta(d_i), \phi(n)) = 1 \\ (p, q, d'_1, ..., d'_{poly(k)}) & \text{where} & d'_i \in_U \mathbb{Z}^*_{\phi(n)} \end{array}$$

are computationally **indistinguishable**, assuming that π_β is pseudo-random. □

Because of the uniformity that appears to be derivable from Theorem 2.2.11, the use of π_β seems to be overkill. This result suggests that a number of bits linear in the size of n, i.e. $2x(k-1)$ for a constant x, might be derivable from the modular inversion of e.

Example 2.2.16 Apply Theorem 2.2.11, with $x = 1/32$. It follows that for any fixed $e \in [2^{(k-1)/8}, 2^{15(k-1)/8}]$ and primes $p, q \in_U [2^{k-1}, 2^k]$, approximately a $x = 1/32$ fraction of the MSB of $e^{-1} \bmod \phi(n)$ are distributed exponentially close to the uniform distribution, with the compared distribution as precisely stated in the theorem.

Instead of fixing e and sampling n, consider the following. Fix n and sample a polynomial number of exponents e_i that satisfy three conditions: they are appropriate RSA exponents in $\mathbb{Z}^*_{\phi(n)}$; they are small enough, i.e. of size at most $k/2$, such that Wiener's Theorem applies; they are in the range of application of Theorem 2.2.11. In other words $e_i \in_U [2^{k/8}, 2^{k/2}] \cap \mathbb{Z}^*_{\phi(n)} \subset [2^{(k-1)/8}, 2^{15(k-1)/8}]$. Then one lets

$D_i = d_i]^{k/16} : \pi_\beta(d_i\rfloor_{31k/16})$, so

$$(p, q, D_1, ..., D_{poly(k)}) \quad \text{where} \quad d_i \equiv e_i^{-1} \bmod \phi(n)$$
$$(p, q, d'_1, ..., d'_{poly(k)}) \quad \text{where} \quad d'_i \in_U \mathbb{Z}^*_{\phi(n)}$$

may be computationally **indistinguishable**, assuming that π_β is pseudo-random.□

The values of D_i cannot be correlated via Wiener's Theorem, as they were in Example 2.2.14, because only the upper bits of the d_i are known, so the continued fraction algorithm (used in the proof of the theorem) is not directly applicable. It is not known whether the e_i being small imply that the MSB of the d_i have a special form.

From Theorem 2.2.11, it appears sensible to conjecture that such modified inverses of e cannot be computationally distinguished from a uniformly and independently distributed set. Note e's different "sampling" interval of $[2^{k/8}, 2^{k/2}] \cap \mathbb{Z}^*_{\phi(n)}$ as opposed to $[2^{(k-1)/8}, 2^{15(k-1)/8}]$ in the theorem. We state this as the following assumption.

Assumption 2.2.17 *Consider a sufficiently large interval of uniform and independent sampling for p and q and a set of size exponential in k of sufficiently large values of e (sampled uniformly within the intersection of $\mathbb{Z}^*_{\phi(n)}$ and of a given range $\mathcal{E}$, which will be made precise). Then, a linear fraction of the upper bits of the inverses of e modulo $\phi(n)$ cannot be computationally distinguished from the equivalent bits in a uniformly and independently distributed set, with the compared distribution as precisely stated in Theorem 2.2.11.*

More precisely, for primes $p, q \in [2^{k-1}, 2^k]$ and for x sufficiently small, if $e_i \in_U \mathcal{E}$, for a significantly large $\mathcal{E} \subset [2^{4x(k-1)}, 2^{(2-4x)(k-1)}] \cap \mathbb{Z}^*_{\phi(n)}$, then any polynomial-time bounded adversary has a negligible probability of distinguishing the following two distributions:

$$(p, q, d_1\rceil^{2xk}, ..., d_{poly(k)}\rceil^{2xk}) \quad \textit{where} \quad d_i \equiv e_i^{-1} \bmod \phi(n) \textit{ with } e_i \in_U \mathcal{E}$$
$$(p, q, d'_1\rceil^{2xk}, ..., d'_{poly(k)}\rceil^{2xk}) \quad \textit{where} \quad d'_i \in_U \mathbb{Z}^*_{\phi(n)}$$

and where $poly(k)$ is a polynomial function in the size of n.

The values of D_i cannot be correlated via Wiener's Theorem, as they were in Example 2.2.14. On one hand, if the e_i are small, then as in Example 2.2.16, the MSB of the d_i are not known to have a special form. On the other hand, if the e_i could be picked so that the d_i were small, then their MSB would have a special form (that is, be all zeros). However, this is avoided by picking the e_i uniformly in a non-interrupted sub-interval of $\mathbb{Z}^*_{\phi(n)}$. In this way, such special values of e_i cannot be favored. This restriction on the ways of picking e_i relates to Assumption 2.2.17$'$.

Also the use of a strong assumption such as that π_β is pseudo-random may not be entirely necessary. Rather, let function π_β be the *exclusive OR* operation with the random string β. It is reasonable to make the hypothesis that π_β composed with modular inversion behaves somewhat like a pseudo-random function. In fact, we require not this function composition to be pseudo-random. We only require that its output be indistinguishable from the uniform distribution, given that the distinguisher knows the factorization of n.

Assumption 2.2.18 *Let $\ell = |\beta|$, let $\pi_\beta : \{0,1\}^\ell \to \{0,1\}^\ell$ be defined as the exclusive OR $\pi_\beta(m) = \beta \oplus m$ and let $g_\ell(d)$ denote some ℓ fixed bits of a variable d. Suppose that $e \in_U \{0,1\}^{2k}$ or, if n is fixed or sampled first, $e \in_U \mathbb{Z}^*_{\phi(n)}$ or e is sampled in an appropriate sub-interval of the preceding, and n is the product of two admissible primes, so that $e \in \mathbb{Z}^*_{\phi(n)}$ after the samplings of n and e. Then for $\beta \in_U \{0,1\}^\ell$, the parameterized distribution defined as*

$$D_{\beta,e,n} = \pi_\beta \circ g_\ell(e^{-1} \bmod \phi(n))$$

is indistinguishable from the uniform distribution on ℓ bits, given that the distinguisher knows the factorization of n.

This is reasonable because xoring does not interact naturally with modular algebraic operators. For instance, the two operations of Assumption 2.2.18 do not satisfy distributivity, neither can their order of application be inverted. Xoring is done with key β and modular inversion, with key e.

Example 2.2.16 can be modified to become the following.

Example 2.2.19 Apply the conditions of Assumption 2.2.17, with $x = 1/32$. For primes $p, q \in [2^{k-1}, 2^k]$ and for $\mathcal{E} = [2^{k/8}, 2^{k/2}] \cap \mathbb{Z}^*_{\phi(n)}$ which is a significantly large subset of $[2^{(k-1)/8}, 2^{15(k-1)/8}]$, the $x = 1/32$ MSB of $e^{-1} \bmod \phi(n)$ are distributed exponentially close to the uniform distribution.

Fix n and sample

$$e_i \ \in_U \ [2^{k/8}, 2^{k/2}] \cap \mathbb{Z}^*_{\phi(n)}.$$

Let $\pi_\beta(m) = \beta \oplus m$ and let $D_i = d_i\rceil^{k/16} : \pi_\beta(d_i\rfloor_{31k/16})$ so that

$$\begin{array}{rrl} (p, q, D_1, ..., D_{poly(k)}) & \text{where} & d_i \equiv e_i^{-1} \bmod \phi(n) \\ & \text{and} & \gcd(D_i, \phi(n)) = 1 \\ (p, q, d'_1, ..., d'_{poly(k)}) & \text{where} & d'_i \in_U \mathbb{Z}^*_{\phi(n)} \end{array}$$

are computationally **indistinguishable**, given Assumptions 2.2.17 and 2.2.18. □

Example 2.2.16 can also be modified so that the set of e (more precisely of inverses of d) is larger and does not contain an obvious weakness, as in Example 2.2.14.

Example 2.2.20 Apply the conditions of Assumption 2.2.17, with $x = 1/32$. For primes $p, q \in [2^{k-1}, 2^k]$ and for $e \in_U [2^{(k-1)/8}, 2^{15(k-1)/8}] \cap \mathbb{Z}^*_{\phi(n)}$, the $x = 1/32$ MSB of $e^{-1} \bmod \phi(n)$ are distributed exponentially close to the uniform distribution.

Fix n and sample $e \in_U [2^{k/8}, 2^{k/2}] \cap \mathbb{Z}^*_{\phi(n)} \subset [2^{(k-1)/8}, 2^{15(k-1)/8}]$. Then let $D_i = d_i\rceil^{k/16} : \pi_\beta(d_i\rfloor_{31k/16})$. Let a uniformly sampled c and other corresponding parameters (d in place of δ and e, of ϵ) satisfy the hypotheses of Theorem 7.1.1. Then

$$\begin{array}{rrl} (p, q, D_1, ..., D_{poly(k)}) & \text{where} & d_i \equiv c e_i^{-1} \bmod \phi(n) \\ & \text{and} & \gcd(D_i, \phi(n)) = 1 \\ (p, q, d'_1, ..., d'_{poly(k)}) & \text{where} & d'_i \in_U \mathbb{Z}^*_{\phi(n)} \end{array}$$

are computationally **indistinguishable**, given Assumptions 2.2.17 and 2.2.18. □

A statement about the modular inverse of any e would be more useful. It is sufficient for our purposes that modular inverses fed into a randomizing function behave like pseudo-random functions.

Definition 2.2.21 *Let $\mathcal{D}$ be a probability distribution on an exponential size set S, and let $f_n : S \to S$ be a keyed function. With the application of f_n, the distribution becomes $f(\mathcal{D})$. Suppose that $f(\mathcal{D})$ is close enough to being evenly spread out in S so that feeding it into a randomizing function makes this function composition behave like pseudo-random function. Then $f(\mathcal{D})$ is said to be* **well pre-shuffled** *in S.*

The statement that a $f(\mathcal{D})$ is well pre-shuffled is weaker than it being indistinguishable from uniformly distributed. A uniform distribution is well pre-shuffled, because one expects a random selection to be evenly spread out in S. A well pre-shuffled distribution is not, in general, uniform, because the f_n may be reversible, for instance if every element of $f(\mathcal{D})$ is spread out exactly evenly in S.

Assumption 2.2.17$'$ *Consider a sufficiently large interval of uniform and independent sampling for p and q and the intersection of $\mathbb{Z}^*_{\phi(n)}$ and of a given range $\mathcal{E}$, forming a set of size exponential in k of values of e. Then the inverses of e modulo $\phi(n)$ are well pre-shuffled in S.*

The supposition of Assumption 2.2.17$'$ implies that the supposition of (an adapted version of) Assumption 2.2.18 makes more sense, denoted as Assumption 2.2.18$'$, which follows. It is coherent that the primed version of Assumption 2.2.17 is stronger and that the one of Assumption 2.2.18 is weaker than the corresponding original one. In addition, the supposition of Assumption 2.2.18$'$ will be more useful.

Assumption 2.2.18′ *Let $\pi_\beta : \{0,1\}^{2k} \to \{0,1\}^{2k}$ be defined as $\pi_\beta(m) = \beta \oplus m$. Suppose that $e \in_U \{0,1\}^{2k}$ or, if n is fixed or sampled first, $e \in_U \mathbb{Z}^*_{\phi(n)}$ or e is sampled in an appropriate sub-interval of the preceding, and n is the product of two admissible primes, so that $e \in \mathbb{Z}^*_{\phi(n)}$ after the samplings of n and e. Then for $\beta \in_U \{0,1\}^\ell$, the parameterized distribution defined as*

$$D_{\beta,e,n} = \pi_\beta(e^{-1} \bmod \phi(n))$$

is indistinguishable from the uniform distribution on $2k$ bits, given that the distinguisher knows the factorization of n.

The two preceding examples are modified similarly.

Example 2.2.19′ Apply the conditions of Assumptions 2.2.17′ and 2.2.18′. For primes $p, q \in [2^{k-1}, 2^k]$ and for $e \in_U \mathbb{Z}^*_{\phi(n)}$, the parameterized distribution $D_{\beta,e,n}$ is indistinguishable from the uniform distribution on $2k$ bits, given that the distinguisher knows the factorization of n. Then

$$\begin{aligned} &(p, q, D_1, ..., D_{poly(k)}) \quad \text{where} \quad && D_i \equiv \pi_\beta\left(e^{-1} \bmod \phi(n)\right) && (2.6)\\ &\quad\text{and} && \gcd(D_i, \phi(n)) = 1 &&\\ &(p, q, d'_1, ..., d'_{poly(k)}) \quad \text{where} && d'_i \in_U \mathbb{Z}^*_{\phi(n)} && (2.7)\end{aligned}$$

are computationally **indistinguishable**. □

Example 2.2.20′ Apply the conditions of Assumptions 2.2.17′ and 2.2.18′. For primes $p, q \in [2^{k-1}, 2^k]$ and for $e \in_U \mathbb{Z}^*_{\phi(n)}$, then $e^{-1} \bmod \phi(n)$ is well pre-shuffled. Let d (replacing δ) and e (replacing ϵ) satisfy the hypotheses of Theorem 7.1.1, and so does c, picked uniformly within the range $|c| \le \gamma n^{-3/4} ed$. Then

$$(p, q, D_1, ..., D_{poly(k)}) \quad \text{where} \quad D_i \equiv \pi_\beta \left(ce^{-1} \bmod \phi(n) \right) \tag{2.8}$$
$$\text{and} \quad \gcd(D_i, \phi(n)) = 1$$

$$(p, q, d'_1, ..., d'_{poly(k)}) \quad \text{where} \quad d'_i \in_U \mathbb{Z}^*_{\phi(n)} \tag{2.7}$$

are computationally **indistinguishable**. □

These properties of RSA can be used to compare the security of $\pi_\beta(ce_i^{-1} \bmod \phi(n))$ with to the one of $\pi_\beta(e_i^{-1} \bmod \phi(n))$. The following assumptions are the types of statements that are useful in Chapters 6 and 7.

Assumption 2.2.22 *Distributions 2.6 and 2.7 are computationally indistinguishable.*

Assumption 2.2.23 *Distributions 2.8 and 2.7 are computationally indistinguishable.*

We would like to show that Assumption 2.2.23 is less strong than Assumption 2.2.22. For this, one may show that if Distributions 2.8 and 2.7 are computationally distinguishable, then Distributions 2.6 and 2.7 are computationally distinguishable. To show this, one would transform instances of the latter pair into ones of the former one, which are distinguishable. However, can a distribution of the form $\pi_\beta(e^{-1} \bmod \phi(n))$ be transformed into one of the form $\pi_\beta(c \cdot e^{-1} \bmod \phi(n))$, for sampled c? This seems unlikely, because the first form is exactly the second, for $c = 1$, and this value may be the most difficult case (the reduction is only for the average case). Furthermore, it seems, in general, impossible to sample c and insert it within the π_β, in order to obtain the sought average case. For this to be feasible, an extra condition on π_β can be met. This is stated as follows, in what we believe is the most general form possible.

Theorem 2.2.24 *Suppose that π_β is such that*

$$\pi_\beta(c \cdot e^{-1} \bmod \phi(n)) = g\left(c, \pi_\beta(e^{-1} \bmod \phi(n))\right)$$

for some function g that may depend on π, but not on β. If Assumption 2.2.22 holds then Assumption 2.2.23 holds.

proof: Suppose that $\mathcal{D}$ distinguishes the two former distributions with probability $1/2+\varepsilon$. Let x be sampled uniformly from Distributions 2.6 or 2.7, and let a uniformly sampled c satisfy the hypotheses of Theorem 7.1.1. Then

$$g(c,x) \in \begin{cases} \text{Distribution 2.8} & \text{if } x \in \text{Distribution 2.6,} \\ \text{Distribution 2.7} & \text{if } x \in \text{Distribution 2.7.} \end{cases} \tag{2.9}$$

In other words, multiplying by a uniform c transforms Distribution 2.6 into Distribution 2.8 and leaves Distribution 2.7 unchanged. This is because, in the first case, $x = \pi_\beta\left(e^{-1} \bmod \phi(n)\right)$, so $g(c,x) = g\left(c, \pi_\beta(e^{-1} \bmod \phi(n))\right) = \pi_\beta(c \cdot e^{-1} \bmod \phi(n))$. Therefore

$$\mathcal{D}'(x) = \mathcal{D}\left(g(c,x)\right)$$

distinguishes Distributions 2.6 and 2.7 with probability $1/2+\varepsilon$. ■

Theorem 2.2.24 is stated to fit with statements proven in the following chapters. Intuitively, the converse statement does not appear to hold. That Distributions 2.8

and 2.7 be computationally indistinguishable may be attributed to the randomness due to c. In other words that Distributions 2.6 and 2.7 be computationally distinguishable is a weaker statement than the sufficient conditions of the previous theorem's proof by contrapositive. It appears to be insufficient to imply distinguishability for the other pair of distributions.

Theorem 2.2.25 (Generalization) *In Theorem 2.2.24, the following can be generalized. The operation $\cdot$ can be changed to any group operation. The distribution of c can be changed to any distribution within the appropriate range.*

One last computational assumption is stated, stronger than any other one in this section, at the exception of Assumption 2.2.18′.

Assumption 2.2.26 *Consider a sufficiently large interval of uniform and independent sampling for p and q and a range of size exponential in k of sufficiently large and admissible values of e. Then, the set of inverses of e modulo $\phi(n)$ cannot be computationally distinguished from a uniformly and independently distributed set, with the compared distribution as precisely stated in Theorem 2.2.11.*

More precisely, similarly as for Assumption 2.2.17, we suppose the following. For primes $p, q \in [2^{k-1}, 2^k]$ and for some x, y, let $e \in_U \mathcal{E}$, for a significantly large subset $\mathcal{E} \subset [2^{2x(k-1)}, 2^{(2x+2y)(k-1)}] \cap \mathbb{Z}^*_{\phi(n)}$. Let

$$D_i = d_i\rceil^{(2-4\alpha)k} : \pi_\beta\left(d_i\rfloor_{(2-4\alpha)k}\right)$$

for a pseudo-random function π_β and $\alpha < 1/6$ but large enough. Then any polynomial-time bounded adversary has a negligible probability of distinguishing the following two *distributions:*

$$(p, q, D_1, ..., D_{poly(k)}) \quad \textit{where} \quad d_i \equiv e_i^{-1} \bmod \phi(n)$$
$$(p, q, d'_1, ..., d'_{poly(k)}) \quad \textit{where} \quad d'_i \in_U \mathbb{Z}^*_{\phi(n)}$$

and where $poly(k)$ is a polynomial function in the size of n.

It is useful in Chapter 7, as it is on this assumption that the security of Algorithm G_4 (Figure 7–9) is based. Denote by Assumption 2.2.26′ the original assumption, along with more complete domain for e (later replaced by δ) and the assumption that the addition of $c\beta$ makes the distribution of the $(2-4\alpha)k$ lower bits indistinguishable from uniform, because c is uniformly distributed.

Assumption 2.2.26′ *Suppose Assumption 2.2.26 with a polynomial size sample of*

$$e \in_U [1, 2^{2\alpha k}] \cap \mathbb{Z}^*_{\phi(n)}$$

and let $d \equiv e^{-1} \bmod \phi(n)$. Then with c uniformly distributed, the parameterized distribution $D_{\beta,e,n}$ defined as

$$D_{\beta,e,n} = \pi_\beta(d) = d + c\beta$$

is indistinguishable from the uniform distribution on $2k$ bits, given that the distinguisher knows the factorization of n.

The second part of the *Discussion* after Algorithm G4 (Subsection 7.1.4) criticizes its application to this security statement and derives an open question.

The RSA permutation. The RSA encryption function is a candidate trapdoor one-way permutation. An efficient pseudo-random generator can be derived from the (assumed) one-wayness of a variant of the RSA function [SPW06].

Under some conditions, the RSA encryption function is a pseudo-random permutation. In this context of this work, the following proposition is often useful. Proposition A.3.6 is the equivalent proposition for EG.

Proposition 2.2.27 *By adding a linear fraction of random bits to a message, m, the* RSA-OAEP *encryption function of m is in practice considered indistinguishable from a uniformly distributed number, unless computing the e^{th} root modulo n is easy.*

proof sketch: This property is related to *semantic security* or *indistinguishability of encryptions* [GM84], usually denoted IND. A concrete way of achieving semantic security is through the RSA-OAEP padding and encryption algorithm provided that the mask generation functions are viewed as random oracles. A detailed description follows this proof sketch.

However, the indistinguishability property sufficient for this work's results is that the output of some encryption function be pseudo-random. This appears to be a property stronger than indistinguishability of encryptions. Nevertheless, Phan and Pointcheval have shown an equivalence between indistinguishability of encryptions and pseudo-randomness [PP04]. ■

That a slightly modified RSA encryption function can be assumed to be a pseudo-random permutation generation is sufficient for this work's results. RSA-OAEP is thus sufficient to justify Proposition 2.2.27. Overall, a slightly modified RSA encryption function is an assumed pseudo-random permutation generator, on which some of this work's results are based on.

For this, the RSA encryption function $f : \{0,1\}^k \to \{0,1\}^k$ is seen as:

$$f : \{0,1\}^{\ell+k_1} \times \{0,1\}^{k_0} \to \{0,1\}^{\ell+k_1} \times \{0,1\}^{k_0}$$

where $k = \ell + k_0 + k_1$.

Let the mask generation functions be hash functions $G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}$ and $H : \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$. The encryption of a message $m \in \{0,1\}^\ell$ with randomness $r \in_U \{0,1\}^{k_0}$ is

$$c = f(s,t)$$

where

$$\begin{aligned} s &= (m{:}0^{k_1}) \oplus G(r) \\ t &= r \oplus H(s) \end{aligned}$$

and the decryption is

$$\begin{aligned} (s,t) &= f^{-1}(c) \\ r &= t \oplus H(s) \\ m &= s \oplus G(r). \end{aligned}$$

2.3 Application issues and standards

In the application of RSA, there are a number of different standard precautions to be taken. The PKCS standard is meant to cover these precautions, for practical purposes. Some of these precautions concern *weak keys*, i.e. keys that are rejected, because they make the cryptosystem behave in ways such that its security may be significantly decreased. This provides a standard set of keys to reject from practical use.

2.3.1 Direct attacks

If some of the cryptosystem's parameters have special values, then the cryptosystem can be vulnerable to direct attacks. An example of vulnerability is when keys are chosen to be in a certain range or of specific values that make them weak keys.

Values	method	effect on security
p or q short, possibly up to 67 digits [Dod06]	elliptic curve factoring method	n factorized
$\phi(n)$ (or rather $p-1$ or $q-1$) has only small prime factors (*)	Pollard $p-1$ Algorithm [MvV01, Section 3.2.3]	n factorized
$p+1$ or $q+1$ has only small prime factors (*)	$p+1$ factoring algorithm [MvV01, p.125, Section 3.12]	n factorized
large prime $r \mid \phi(n)$ is s.t. $r-1$ has only small prime factors (*)	cycling attacks [MvV01, Section 8.2.2(vii)]	n factorized
$p-q \in \mathcal{O}(n^{1/4})$	Fermat's algorithm	n factorized

Table 2–1: Consequences on RSA security of values of p and q. When the (*) conditions are avoided, p and q are said to be *strong primes*.

Strong primes may be generated with an algorithm given in the *Handbook of Applied Cryptography* [MvV01, Section 4.4.2].

Values	method	effect on security
$d < n^{0.25}/3$	Wiener [Wie90]	n factorized
$d < n^{0.292}$	Boneh and Durfee [BD00]	n factorized
small e (low-exponent RSA) under different public keys only	Håstad [Hås88]	message, m, recovered
small e (low-exponent RSA)	Coppersmith et al. [CFPR96]	m recovered
$ed \equiv c \bmod \phi(n)$ such that $d \leq n^{0.25}/3$ and $\lvert c\rvert \leq \gamma n^{0.75} ed$	Blömer and May [BM04]	n factorized

Table 2–2: Consequences on RSA security of values of e and d.

Wiener's method is cited in this work as Theorem 6.3.1. Blömer and May's method is cited in this work as Theorem 7.1.1.

2.3.2 Indirect attacks

Well-known indirect attacks on RSA may come from the knowledge of additional parameters which are not public nor private, but yield the private ones.

Indirect attacks may also be made possible from additional interaction with the encrypting or decrypting parties.

2.4 Chapter notes

This chapter is mostly based on Stinson's book [Sti06, Chapter 5].

Subsection 2.2.3 contains material that has been updated in the most recent standard. In PKCS#1 v2.1 $\phi(n) = (p-1)(q-1)$ is changed to $\lambda(n) = \mathrm{lcm}(p-1, q-1)$

Values	method	effect on security
$\phi(n)$	solve a second degree equation (knowing n)	n factorized
$k \cdot \phi(n)$	[Mil75, Lemma 4]	n factorized
part of d	Boneh, Durfree and Frankel [BDF98]	n factorized
part of m	Håstad and Näslund [HN98]	m recovered

Table 2–3: Consequences on RSA security of additionally known parameters.

Interaction	effect on security
chosen ciphertext attack: an oracle of e^th root extraction (e.g. man-in-the-middle attack, Figure 1–2)	unknown properties [PKC03, Section 7.1]
key generation with verifiable randomness	Juels and Guajardo [JG02]

Table 2–4: Consequences on RSA security of additional interaction.

[PKC03, Section 3]. Because $\lambda(n) \mid \phi(n)$, this allows to maintain the properties to be developed in Subsection 2.2.4 while ridding $\phi(n)$ of some redundant factors, as $\lambda(n) = \phi(n)/\gcd(p-1, q-1)$. This allows for a lesser number of modular inverses to be rejected, because $|\mathbb{Z}_{\lambda(n)}| < |\mathbb{Z}_{\phi(n)}|$. However, fewer are accepted as well, because $\phi(\lambda(n)) < \phi(\phi(n))$, and therefore $|\mathbb{Z}^*_{\lambda(n)}| < |\mathbb{Z}^*_{\phi(n)}|$. The values of primes to be avoided, as listed in Table 2–1, still apply.

The non-asymptotic bounds in Theorem 2.2.4 are form the Handbook of Applied Cryptography [MvV01, Fact 2.96].

The information in Subsection 2.2.6 is not explicit in Stinson's book, but can easily be derived from the mentioned chapter.

Subsection 2.2.8 is original work based on Theorem 2.2.11. At the end of this section, the RSA-OAEP padding and encryption algorithm can be referred to in [FOPS01, PKC03].

In Section 2.3, Fermat's factorization algorithm of Table 2–1 may be referred to in [Wei07].

CHAPTER 3
RSA signature padding

3.1 Introduction

Background. In [CKN00], Coron, Koeune and Naccache present security reductions for RSA signature padding schemes. These reductions allow to go from fixed-length messages to arbitrary-length messages RSA signature padding schemes. A hash function μ is an atomic primitive that is assumed to be a secure padding scheme for RSA. However, μ takes a $2k + 1$ bit input and returns a $2k$ bit output where $2k$ is the length of the RSA modulus. This particularity of the scheme is not significantly modifiable: the bit length of the μ output has to have about the same bit length as the RSA modulus. This limitation on the choice of μ forces either to instantiate it with a non-dedicated hash function, or with a dedicated hash function that uses both compression and chaining primitives.

New algorithm. In this chapter, with a similar construction, we give a practical instantiation based on the compression function of SHA-1 without any chaining function. Our solution has the great advantage over [CKN00] of removing the relation of the length of μ output to the length of the RSA modulus. We are able to achieve this result simply by making an additional assumption about μ, namely division intractability. This property is slightly stronger than collision intractability.

The new algorithm allows one to make precomputations on partially received messages. For instance, IP packets are typically received in a random order.

3.1.1 Hash-and-sign paradigm

A common practice for signing with RSA is known as the hash-and-sign paradigm. First, a hash or redundancy function, which usually consists of a compression function and a chaining function, is applied to the message. Then some padding is added to the result, and this value is exponentiated using the signature exponent. This is the basis of several existing standards many of which have been broken, as surveyed by [Mis98].

Paradigm 3.1.1 (Hash-and-sign paradigm) *A signature algorithm pre-processes a message to be signed by padding it as follows.*

1. *It is more efficient to exponentiate a shorter, hashed, number. The security then also depends on the one of the hash function μ. More formally, let μ be a randomized compression function taking as input a message of size t, using r random bits, and outputting a message digest of length l:*

$$\mu : \{0,1\}^r \times \{0,1\}^t \rightarrow \{0,1\}^l$$

2. *The hashed message is also padded via an encoding function enc in order to use the full length of the* RSA *modular exponentiation. Let enc be an encoding function taking as input a message digest of size l, and outputting an encoded message of length $2k$:*

$$enc : \{0,1\}^l \rightarrow \{0,1\}^{2k}$$

Overall:

$$enc \circ \mu : \{0,1\}^r \times \{0,1\}^t \rightarrow \{0,1\}^{2k}$$

Firstly, sum up the mathematical consideration of the RSA signing function. The RSA signing function, as conceived from theoretical (abstract) point of view to a progressively more applied (implementable) one, is illustrated by Figure 3–1.

$$\begin{tikzcd}
\boxed{Sign(y) = (y^d \bmod n, y)} \arrow[d] \\
\boxed{Sign(y) = (\mu(y)^d \bmod n, y)} \arrow[d] \\
\boxed{Sign(y) = (enc \circ \mu(y)^d \bmod n, y)}
\end{tikzcd}$$

Figure 3–1: The RSA signing function, abstracted form to more applied form. The top line is the most abstracted form: only signing by exponentiation is shown. The middle line includes the hashing step, with the hash function μ. The bottom line is the most applied one: an additional step of encoding compensates for limitations in the lengths of the hash function.

The *enc* function. For clarity and completeness, we briefly elaborate on the *enc* function. Bleichenbacher showed an efficient attack based on a feature of the PKCS # 1 standard for hashing-and-signing [Ble98]. This feature is that some of the bits generated by *enc* are constant. A way to avoid this is to use a randomized padding function (it is preferable to avoid using memory as a way to remember which messages have already been signed). Also, if *enc* is deterministic, different applications of the encoding operation to the same message produce the same encoded messages, which is undesirable [PKC03, Section 9]. In practice, this yields tighter security proofs, for instance in Bellare and Rogaway's paper that randomizes FDH to obtain PSS [BR96] and [PKC03, Section 8.1]. Therefore, the padding added via *enc* is required to be pseudo-random and *enc* is, to be a randomized fonction.

Secondly, consider an algorithmic version of the RSA key generation and signing function. Figure 3–2 shows the pseudo-code of the classical RSA signature scheme $(Gen, Sign, Verify)$ which signs fixed-length t-bits messages. This figure is a modification of [CKN00, Figure 1].

```
SYSTEM PARAMETERS
    an integer $k > 0$
    a function $\mu : \{0,1\}^r \times \{0,1\}^t \to \{0,1\}^l$
    a function $enc : \{0,1\}^l \to \{0,1\}^{2k}$
KEY GENERATION : $Gen$
    $(n, e, d) \leftarrow \text{RSA}(1^{2k})$
    public key: $(n, e)$
    private key: $(n, d)$
SIGNATURE GENERATION : $Sign$
    $R \leftarrow_U \{0,1\}^r$
    $m \in \{0,1\}^t$
    $y \leftarrow enc \circ \mu(R, m)$
    return $\langle R, y^d \bmod n \rangle$
SIGNATURE VERIFICATION : $Verify$
    $y \leftarrow x^e \bmod n$
    $y' \leftarrow enc \circ \mu(R, m)$
    if $y = y'$ then return 1 else return 0
```

Figure 3–2: The classical RSA scheme using $enc \circ \mu$ for signing fixed-length messages.

3.1.2 Chosen-ciphertext attacks

Therefore, a particular class of attack on RSA targets the security of the padding algorithm. These attacks include chosen-ciphertext attacks (CCA), as shown in Figure 3–3. They are a special case of a man-in-the-middle (MITM) attack, where the adversary has access to a signature oracle.

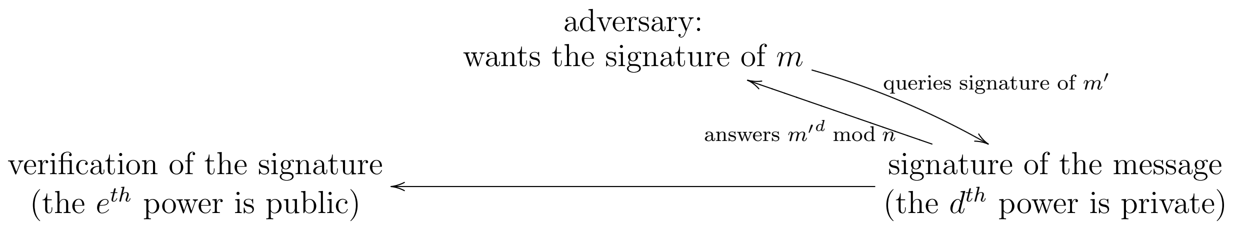

adversary:
wants the signature of m
queries signature of m'
answers ${m'}^d \bmod n$
verification of the signature
(the e^{th} power is public)
signature of the message
(the d^{th} power is private)

Figure 3–3: Chosen-ciphertext attacks (CCA) on a signature algorithm: the signing algorithm acts as a signature oracle for the adversary. If the queries are not chosen all at once, the figure can be seen as showing an *adaptive* version of this attack, where the adversary can adapt its strategy w.r.t. the oracle's previous answers.

The RSA cryptosystem used without the hash-and-sign paradigm is subject to a well-known CCA [Dav82]. Suppose that the adversary is interested in knowing the signature of a message m, that is,

$$m^d \bmod n.$$

The adversary can query the signature of a random, innocent-looking message,

$$m' = s^e m \bmod n,$$

for a random integer s. The decryption oracle returns the signature of m':

$$(s^e m)^d \bmod n \equiv s m^d \bmod n.$$

This signature, multiplied by $s^{-1} \bmod n$ provides the required signature.

In practice, a CCA can be deployed as a *lunch time* attack, where the adversary has access to a signing device at a time where its legitimate user leaves it untended. A concrete example is the one of a smart card which may be under the full control of the adversary.

3.1.3 Security goal

The security of signature schemes was formalized in [GMR88] for the asymptotic setting. We will prove security against existential forgery by adaptive chosen-message adversaries [1] . Overall, we show that adaptive chosen-message attack cannot produce existential forgeries.

The adaptive chosen-message attack is the most general attack, that is, the one involving the strongest adversary amongst the standard theoretical attacks catalogued in [GMR88]. As in Figure 3–3, the adversary can use the signature algorithm as an oracle. The adversary can request signatures of messages that depend on the given public key. Additionally, the adversary can request signatures of messages that depend on previously obtained signatures. Similarly, the requests can depend on the target message to be forged.

The least success that an adversary may hope for, again amongst the standard interesting adversarial results, is an existential forgery. To forge a signature means to produce a new signature that was not requested to the oracle. An existential forgery is a forgery for at least one message, so that the adversary may not control which message it forges a signature for, so the message may be random or nonsensical. Therefore, such an adversary may not be able to do more than minor damages. Note

[1] In the previous section, the message is called ciphertext because it is sent to a signature oracle, which is also, by abuse of language, called a decryption oracle. This is because of the common convention that the RSA private key is the decryption key.

that Figure 3–3 does not show an existential forgery, but a, more severe, selective forgery: the forgery is for a message chosen a priori by the adversary.

3.2 Definitions

The following definitions are for the exact security of signature schemes [BR96].

Definition 3.2.1 *A forging algorithm F is said to* $(t_{proc}, q_{sign}, \varepsilon)$-break *the signature scheme given by* $(Gen, Sign, Verify)$ *if after at most q_{sign} adaptively chosen signature queries and t_{proc} processing time, it outputs a valid forgery with probability at least ε. The probability is taken over the random bits of F, and given that the random bits in the signature, R, are correctly distributed.*

Definition 3.2.2 *A signature scheme* $(Gen, Sign, Verify)$ *is* $(t_{proc}, q_{sign}, \varepsilon)$-secure *if there is no forging algorithm which $(t_{proc}, q_{sign}, \varepsilon)$-breaks the scheme.*

To construct μ from a dedicated hash function without chaining, we make an additional assumption, which is strong but constructible. We use a definition of [GHR99] slightly modified for our purposes.

Definition 3.2.3 *Let $negl(l)$ be a negligible function and let H_l be a collection of compression functions that map strings of length t into strings of length l. Such a collection is said to be* $negl(l)$-division intractable *if for $\mu \in H_l$, it is infeasible to find distinct inputs $X_1, ..., X_n, Y$ such that $\mu(Y)$ divides the product of the $\mu(X_i)$'s.*

We modify this definition in three ways. Firstly, it is also required that it is infeasible to find distinct inputs $X'_1, ..., X'_n$ such that $\mu(Y)$ divides the inverse of the product of the $\mu(X_i)$'s. Secondly, the products are taken modulo 2^t. Thirdly, A is a probabilistic algorithm and a portion of the input length of μ may be randomized.

Formally, for every probabilistic algorithm A that runs in polynomial time in l, there exists a l_o such that for all $l \geq l_o$ and for all $Y, \{X_i'\}_i$ and $\{X_i\}_i$:

$$P_{\mu \in H_l, \mu\text{'s random bits}} \begin{bmatrix} A(\mu) = \langle X_1, ..., X_n, X_1', ..., X_{n'}', Y \rangle \\ s.t.\ Y \neq X_i,\ i = 1, ..., n,\ \text{and}\ Y \neq X_i',\ i = 1, ..., n', \\ \mu(Y)\ \text{divides}\ \prod_{i=1}^{n} \mu(X_i) \bmod 2^t, \\ \text{and}\ \mu(Y)\ \text{divides}\ \left(\prod_{i=1}^{n} \mu(X_i')\right)^{-1} \bmod 2^t \end{bmatrix} < negl(l)$$

If μ is randomized, the adversary A can choose the input but not the randomness, so that the latter is picked from the uniform distribution. Given a randomly chosen function μ from H_l, A needs to find pairs

$$(R_1, X_1), ..., (R_n, X_n), (R_1', X_1'), ..., (R_{n'}', X_{n'}'), (R, Y)$$

such that $Y \neq X_i$ for $i = 1, ..., n$ and $Y \neq X_i'$ for $i = 1, ..., n'$, but $\mu(R, Y)$ divides the products $\prod_{i=1}^{n} \mu(R_i, X_i) \bmod 2^t$ and $[\prod_{i=1}^{n} \mu(R_i', X_i')]^{-1} \bmod 2^t$.

Remark 3.2.4 *In practice, this version of division intractability means that it is infeasible to find two products of values of μ that are equal. As before, let A choose the input but not the randomness. Given a randomly chosen function μ from H_l, A needs to find pairs*

$$(R_1, X_1), ..., (R_n, X_n), (R_1', X_1'), ..., (R_{n'}', X_{n'}'), (R, Y)$$

such that

$$\mu(R, Y) \prod_{i=1}^{n} \mu(R_i', X_i') \bmod 2^t = \prod_{i=1}^{n} \mu(R_i, X_i) \bmod 2^t$$

but $Y \neq X_i$ for $i = 1, ..., n$ and $Y \neq X_i'$ for $i = 1, ..., n'$.

3.3 An improved algorithm

We construct in Figure 3–4 a new signature scheme $(Gen', Sign', Verify')$ using the function $enc \circ \mu$. The new construction allows the signing of messages of size $2^a(t-a)$ bits where a is between 0 and $t-1$. This is a modification of [CKN00, Figure 2].

```
SYSTEM PARAMETERS
    an integer $k > 0$
    an integer $a \in [0, t-1]$
    a function $\mu : \{0,1\}^r \times \{0,1\}^t \rightarrow \{0,1\}^l$
    a function $enc : \{0,1\}^l \rightarrow \{0,1\}^{2k}$
KEY GENERATION : $Gen'$
    $(n, e, d) \leftarrow rsa(1^{2k})$
    public key: $(n, e)$
    private key: $(n, d)$
SIGNATURE GENERATION : $Sign'$
    Split the message $m$ into $b$ blocks of size $t-a$ bits
    such that $m = m[1]||...||m[b]$
    $R_i \leftarrow_U \{0,1\}^r$ for $i = 1, ..., b$
    $\alpha \leftarrow \prod_{i=1}^{b} \mu(R_i, i||m[i]) \bmod 2^t$
    where $i$ is the $a$-bit string representing $i$
    $R \leftarrow_U \{0,1\}^r$
    $y \leftarrow enc \circ \mu(R, \alpha)$
    return $\left\langle R, \langle R_i \rangle, y^d \bmod n \right\rangle$
SIGNATURE VERIFICATION : $Verify'$
    $y \leftarrow x^e \bmod n$
    $\alpha \leftarrow \prod_{i=1}^{b} \mu(R_i, i||m[i]) \bmod 2^t$
    $y' \leftarrow enc \circ \mu(R, \alpha)$
    if $y = y'$ then return 1 else return 0
```

Figure 3–4: The new construction using $enc \circ \mu$ for signing long messages.

Theorem 3.3.1 *For a non-negligilbe function $\varepsilon(l)$ and all negligible $negl(l)$ functions, it holds that $\varepsilon(l) > negl(l)$ for sufficiently large l. Suppose that q_{sign} and t_{proc} are polynomials in l. For a fixed ε, if the signature scheme $(Gen, Sign, Verify)$ is $(t_{proc}, q_{sign}, \varepsilon)$-secure and if μ is negl-division intractable, then the signature scheme described in Figure 3–4 $(Gen', Sign', Verify')$ is $(t'_{proc}, q_{sign}, \varepsilon)$-secure, where:*

$$t'_{proc} = t_{proc} - 2^a \cdot q_{sign} \cdot \mathcal{O}\left(t^2\right)$$

for a message size of $2^a(t-a)$ bits and μ that has a relatively negligible time complexity.

proof: Suppose there is a forger F' that $(t'_{proc}, q_{sign}, \varepsilon)$-breaks the signature scheme $(Gen', Sign', Verify')$. Then, we can construct a forger F that $(t_{proc}, q_{sign}, \varepsilon)$-breaks the scheme $(Gen, Sign, Verify)$ using F'. The forger F has oracle access to a signer S for the scheme $(Gen, Sign, Verify)$ and its goal is to produce a forgery for $(Gen, Sign, Verify)$.

The forger F executes F' and, if F' has queries, then F computes their answers and returns them to F'. Specifically, F executes the steps of $Sign'$ of Figure 3–4 and calls S as a sub-routine. In particular, when F' needs the signature of the j^{th} message m^j, F queries S to obtain the signature s_j of α_j. This takes time $\sum_{j=1}^{q_{sign}} b_j \leq q_{sign} \cdot 2^a$, multiplied by the time necessary to compute multiplications modulo 2^t, which is feasible in time at most quadratic in t. The factor 2^a comes from that the block index has a bits, therefore its value is bounded above by 2^a.

Eventually, F' outputs a forgery (m', s') for the scheme $(Gen', Sign', Verify')$, where $s' = \left\langle R', \langle R'_i \rangle, y'^d \bmod n \right\rangle$. So F can compute:

$$\begin{aligned} y' &= enc \circ \mu(R', \alpha') \\ \alpha' &= \prod_{i=1}^{b'} \mu(R'_i, i || m'[i]) \bmod 2^t \\ b' &= \text{number of blocks of } m' \end{aligned}$$

which, similarly, takes additional time $b' \in 2^a \cdot \mathcal{O}(t^2)$.

The total time complexity of F is the time to simulate the oracle plus the time to compute these last values. The operations other than the multiplications modulo 2^t have relatively negligible time complexities, given that μ has a relatively negligible time complexity. Therefore, the total time is of order $2^a \cdot q_{sign} \cdot \mathcal{O}(t^2)$.

We distinguish two cases:

First case: $\alpha' \notin \{\alpha_1, ..., \alpha_{q_{sign}}\}$. In this case, F outputs the forgery $(y', s'[3])$, where $y' = enc \circ \mu(R', \alpha')$ and $s'[3] = y'^d \bmod n$, the third component of s'. This is a valid forgery for the signature scheme $(Gen, Sign, Verify)$ since y' was never signed by the signer S. This value was never signed, because μ is a hash function on a new value α', so standard assumptions on the security of μ imply that the resulting y' is new, except with negligible probability. Finally, that such a forgery is possible contradicts our assumption that the signature scheme is secure.

Second case: $\alpha' \in \{\alpha_1, ..., \alpha_{q_{sign}}\}$, so there exists a c such that $\alpha' = \alpha_c$. Let us denote $m = m^c$, $\langle R_i \rangle = \langle R^c_i \rangle$, $\alpha = \alpha_c$, and $b = b_c$. We have:

$$\prod_{i=1}^{b'} \mu(R'_i, i || m'[i]) \bmod 2^t = \prod_{i=1}^{b} \mu(R_i, i || m[i]) \bmod 2^t. \tag{3.1}$$

Note that $x \,|\, y \bmod 2^t$ means that there exists a z such that $xz \equiv y \bmod 2^t$. For such μ and z, it is difficult to find a $\bar{z}$ such that $z = \mu(\bar{z})$.

For $m' \neq m$, there are three subcases. Suppose $b' = b$. Then because $m' \neq m$, for some j it holds that $j||m'[j] \notin \{1||m[1], ..., b||m[b]\}$ and $\mu(R'_j, j||m'[j])$ divides the products $\prod_{i=1}^{b} \mu(R_i, i||m[i]) \bmod 2^t$ and $\left(\prod_{i=,i\neq j}^{b} \mu(R_i, i||m[i])\right)^{-1} \bmod 2^t$. Suppose that $b > b'$, then there is at least one more block on the LHS of the previous equation that similarly divides the RHS. The situation is symmetric for $b < b'$. In any of the three cases, since $\varepsilon > negl$, and t' and q_{sign} are polynomial in l, this contradicts our assumption that μ is $negl$-division intractable.

If $m' = m$, then there exist choices of random bits, $R'_i \neq R_i$ for some i, such that Equation 3.1 holds. This yields the same three subcases as above, contradicting division intractability, which is consistent with [PKC03, Section 8.1]: randomness is useful, but not *critical* to security. For simplicity, the random bits picked by F are assumed to be chosen honestly, i.e. as close as possible to uniformly distributed. ■

3.4 Further developments

3.4.1 A practical hashing family $H_{3\cdot 160}$.

The original result was formulated in terms of version 1.5, even though this standard does not use randomization in *enc*. The randomized *enc* function of version 2.1's more robust RSASSA-PSS (see Chapter notes) can more readily be interpreted as a particular case of the previous sections' *enc* function. Using version 2.1 is consistent with this chapter's security proof which is based on the one of the former, i.e. PSS, in [BR96]. Section 3.1.1 explained how a randomized security proof is preferable.

We define the function μ by using the following standard dedicated primitives:

$$\begin{aligned} h &= SHA-1 &&: \{0,1\}^{512} \rightarrow \{0,1\}^{160} \\ enc &= PKCS\ \#1\ ver.\ 2.1 &&: \{0,1\}^{480} \rightarrow \{0,1\}^{2k} \end{aligned}$$

where $|n| = 2k$ and where $enc = PKCS\ \#1\ ver.\ 2.1$ is a feasible randomized instantiation using the parameter for the intended length in bits of an encoded message, $emBits = 480$. In brief, this standard is as follows. A leading 0 bit ensures that the encoded message to be exponentiated is, when converted to an integer, less than the modulus. The encoding's last eight bits are a *trailer field*, formed by the octet 0xbc. The rest of the blocks are pseudo-randomly generated. This makes the length of the encoded message to be exponentiated equal to $2k$. [PKC03, Section 9.1]

Let R_i be a uniformly distributed $2 \cdot 160$-bit string and m_i the i^{th} block of m, the message to be signed. Then μ is the compression function derived by [GHR99, Section 6] from the heuristic given in [BP97]:

$$\mu(R_i, m_i) = 2^{320} \cdot h(m_i) + R_i$$

where h is a collision-intractable function. Also, R_i is the smallest integer greater than the integer defined by the i^{th} 320-bit block such that $\mu(R_i, m_i)$ is odd.

Note that our μ is denoted h in [BR96, Figure 1]'s notation. In that notation, the functions g_1 and g_2 not only produce the padding, but also mask the random bits used in the compression function. This masking is left out in this chapter's scheme, but such a modification is compatible: in fact, it allows the signature generation to skip the transmission of $R, \langle R_i \rangle$, as was done in Figure 3–4.

The function μ is defined only when $\mu(R, m)$ is an odd integer. This guarantees invertibility modulo 2^t. Overall:

$$\begin{aligned} \mu &: \{0,1\}^{320} \times \{0,1\}^{512} \rightarrow \{0,1\}^{480} \\ enc \circ \mu &: \{0,1\}^{320} \times \{0,1\}^{512} \rightarrow \{0,1\}^{2k} \end{aligned}$$

3.4.2 Improved communication complexity.

The signature scheme $(Gen', Sign', Verify')$ described in Section 3.3 has significant overhead communication complexity: the number of random bits transmitted is proportional to the number of message blocks. This problem represents the main open problem of this chapter. This is only an issue when the scheme is considered in general, that is, for an encoding function that does not feature recoverable randomness, as opposed to the one described in Section 3.4.1.

A first solution to this open problem can be sketched using a (conjectured) pseudo-random number generator. The following definition is based on [Lub96, p.50-51].

Definition 3.4.1 *Let $g : \{0,1\}^r \rightarrow \{0,1\}^{(b+1)r}$ be a **P**-time function. We say g is a (δ, t)-*secure pseudo-random number generator *for which adversary A has success probability:*

$$\delta_A = \left| P_{X \in \{0,1\}^r} [A(g(X)) = 1] - P_{Z \in \{0,1\}^{(b+1)r}} [A(Z) = 1] \right|$$

if every adversary A has a probability of success $\delta_A \leq \delta$ and a running time of at least t.

The modified scheme would involve the transmission of r random bits, which would be stretched into $(b+1)r$ random bits via a pseudo-random number generator g, by both the signer and the verifier. The pseudo-random bits take the place of their random counterparts in the original scheme described in Figure 3–2. The security of this modified scheme is implied by the one of the original scheme, and of the pseudo-random number generator.

For the practical implementation described in the previous section (but used with a general encoding function that may not feature recoverable randomness),

$$\mu(R_i, m_i) = 2^{320} \cdot h(m_i) + R_i$$

where h is collision intractable.

Collisions are unlikely. On average over all possibilities, R_i is incremented by a value of 0.5. Therefore, when a collision in μ occurs, it very likely comes from a collision in h, which is in turn unlikely, because h is assumed to be collision intractable.

To quicken the verification process, R_i can be defined as i^{th} 320-bit block + inc_i. In such a case, only the value of inc_i is transmitted with the message block trading time for communication and time of the receiver. This is generally accepted to provide the same distribution, in practice.

3.5 Summary

In [CKN00], the problem of designing a secure general-purpose padding scheme was reduced to the problem of designing a one-block secure padding scheme by providing an efficient and secure tool to extend the latter into the former. By modifying their construction for arbitrary-length messages, and adding one reasonable computational assumption, we provide a practical method of instantiating the secure padding function for short messages using the compression function of dedicated hash functions as well as dedicated encoding functions. We have presented an implementation that uses SHA-1 and PKCS #1 ver. 2.1. This implementation is independent of the size of the RSA modulus. This was not true in [CKN00].

Dedicated hash functions usually consist of two primitive functions, one of compression and one of chaining. This chapter presents an improvement on practicality, since it reduces the potential attacks on the one-block padding scheme to the ones on the hash function's compression function, eliminating all worries about the chaining function, or its interactions with the compression function.

3.6 Chapter notes

This chapter was published as a co-authored paper with Jean-Marc Robert [AR01].

Update in PKCS versions. The latest PKCS document, version 2.1, states the following [PKC03, Section 8]. *Two signature schemes with appendix are specified in this document:* RSASSA-PSS *and* RSASSA-PKCS*1-v1_5. Although no attacks are known against* RSASSA-PKCS*1-v1_5, in the interest of increased robustness,* RSASSA-PSS *is recommended for eventual adoption in new applications.* RSASSA-PKCS*1-v1_5 is included for compatibility with existing applications, and while still appropriate for new applications, a gradual transition to* RSASSA-PSS *is encouraged.*

CHAPTER 4
Introduction to backdoors

The goal of this chapter is to first develop an intuition of the concept of backdoor and then to provide a formal definition. This forms a basis for the chapters that follow. These chapters develop a classification of existing backdoors and provide improvements on some of them.

4.1 Background

Key generators with backdoors can serve two purposes:

1. a designer may distribute a malicious key generation algorithm producing public and private key pairs which are such that a public key allows this attacker to retrieve the corresponding private key;
2. a legitimate key escrow-like system can be based on a backdoored key generator that allows the trusted party to retrieve the private key associated with a public key without an expensive database.

4.1.1 The two contexts

Illegitimate backdoors. Legitimate users of cryptography rely on hardware or software to generate their keys. Therefore, users ultimately rely on smart card manufacturers, software companies (that provide web browsers) or their own capabilities to review open source software in order to assure the validity of the key generation.

For instance, malicious key generators may not use a trustworthy source of randomness. The randomness may be made so that it has a special structure so that weak keys are produced. For the designer of the generator, weak keys are interesting for instance when their private part can be computed from the corresponding public part.

Legitimate backdoors. Once encrypted, a message is unreadable for a party who does not have the decryption key. This is problematic if the legitimate recipient of the message has lost his key. In parallel, law enforcement agencies may wish to have access to the contents of encrypted messages. To address these problems, the mechanism of *key escrow*[1] was developed to recover the key associated with a message.

In a corporate setting, technical support is expected to help legitimate users to retrieve such information. A key escrow generally implemented as a database of keys, as illustrated in Figure 4–1. An example of an implementation is Entrust [Ent] (which they deem *key backup* instead of escrow to avoid legal issues).

In a law enforcement setting, the goal of key escrow is to help with law enforcement. A key escrow allows a third party (such as a government agent) to obtain decryption keys in order to access encrypted information. An example of a hardware implementation is the NSA's Skipjack algorithm [RSAc], where a chip's *unit key* and

[1] In Law, *escrow* is an arrangement by which something is kept in waiting by a third party, that which can be money, property, or source code, until a specified condition is fulfilled. Its etymology is the medieval Latin *scroda* for "scroll", via the Old French *escroe*. The Latin word is of Germanic origin and is related to "shred".

serial number are escrowed. When a Skipjack chip is used for encryption, the encrypted message is appended with its encryption key, itself encrypted with the unit key, and its own serial number, so that the key escrow (actually, two key escrows authorities) may decrypt the encryption key and provide it to the third party.

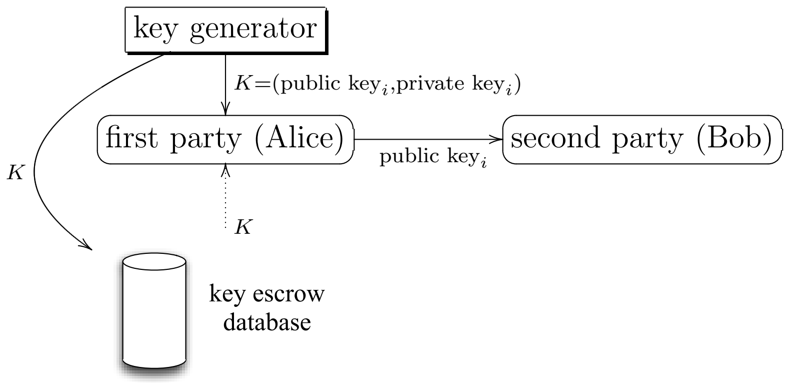


Figure 4–1: High-level view of the roles played by two parties interacting with an escrow.

Definition 4.1.1 *In a public-key cryptosystem,* **key escrow** *is a feature by which decryption keys are held in escrow by a trusted third party. The keys can later be recovered by a legitimate party, which can be the legitimate owner or a law enforcement agency.*

Key escrow purposes. The following parties can play the role of the escrow in a key escrow cryptosystem.

1. Auto-escrow: to recover lost keys. In this case, the escrow may or may not be a distinct third party.
2. Government: to decrypt messages suspected to be relevant to law enforcement. In this case, the escrow authority is probably a government agency.

Even though it will not be addressed in this work, it is to be noted that security issues exist with respect to key escrow methods.

Key escrow alternatives. The simplest way to contaminate a cryptographic black box with a backdoor is to hard code the private values that the designer wishes to obtain and to escrow copies of them. The key which the designer then needs to keep consists of all these escrowed copies. Therefore, the designer requires a database which size is a function of the number of escrowed copies. Any contemporary cryptosystem should have a number of possible keys that is exponential in function of its security parameter $k : |\{k_{pub}\}| \in EXP(k)$ possible legitimate user (distinguisher) keys.

An alternative to key escrow databases is the use of *backdoored key generators.* From a separate piece of private information, i.e. a single key, a cryptosystem's private parameters can be recovered. The amount of memory required to use a backdoor is the one taken by the designer's secret key (symmetric backdoors) or public key (asymmetric backdoors). Therefore, it is a constant amount of space, independent of the number of generated backdoored keys. To sum up, the designer has a space advantage in preferring backdoored keys over escrow.

4.1.2 Intuition on the roles of the parties

In public-key cryptosystems, we usually look at the interaction between two parties who are exchanging messages. Suppose that one calls the first party, Alice, the one for which the keys were generated. Then, the second party, Bob, encrypts messages and the first one decrypts them. Figure 2–3 illustrates how the keys are generated and provided, before a message is signed by Alice or a message is encrypted by Bob.

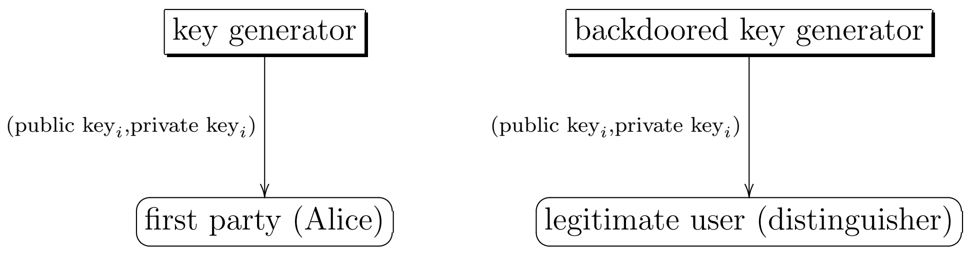

key generator
(public key$_i$,private key$_i$)
first party (Alice)
backdoored key generator
(public key$_i$,private key$_i$)
legitimate user (distinguisher)

Figure 4–2: Change of context: when the key generator produces backdoored keys, Alice becomes the distinguisher in the analysis.

Consider another context. Take this process, but disrupted by a dishonest key generator, as in Figure 4–2. This is a different disruption from the eavesdropper's, in Figure 2–3.

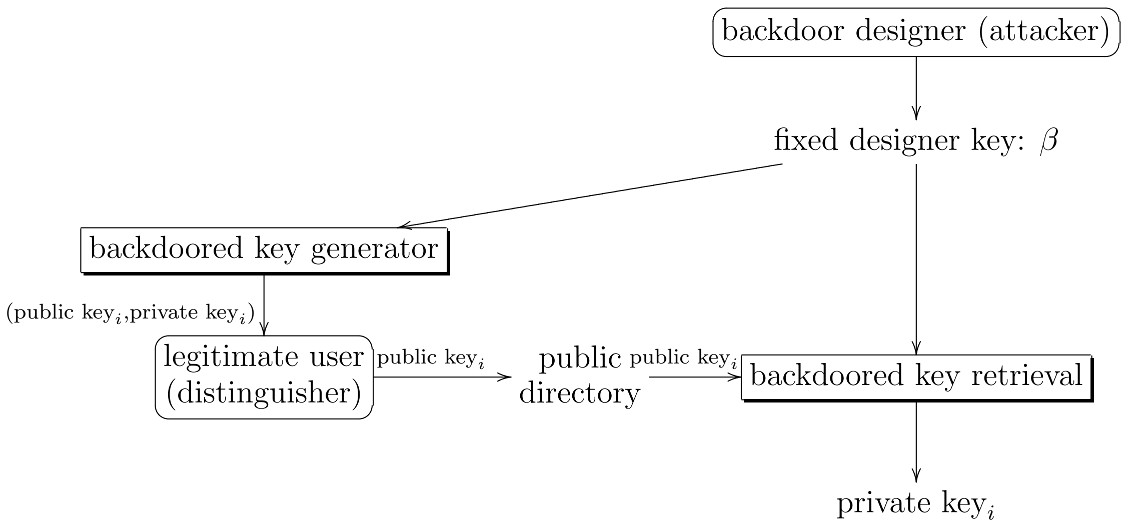

backdoor designer (attacker)
fixed designer key: β
backdoored key generator
(public key$_i$,private key$_i$)
legitimate user (distinguisher)
public key$_i$
public directory
public key$_i$
backdoored key retrieval
private key$_i$

Figure 4–3: High-level view of the two components of a backdoored key generator: the generator and the attack. Compare with the more imageful, equivalent, Figure 1–4.

This key generator is designed to produce *backdoored keys*, i.e. public keys that contain a backdoor. Such a backdoored key allows the designer to easily compute the corresponding private key, as in Figure 4–3. The quality of the backdoored key generator is measured with respect to the adversarial *distinguisher*.

4.1.3 History

Backdoors are a special case of subliminal channels, which study begins in the 1980's. In 1983, Simmons [Sim83, Sim85] introduced the notion of subliminal (covert) channel, by which information is exchanged secretly in innocent-looking messages.

One research area that Simmons' work initiated is the one of information hiding, or steganography. The aims of steganography are the hidden transmission of messages, mostly for intellectual property protection and sometimes for subliminal communication. The channels used are mostly media digital formats and sometimes implemented software code.

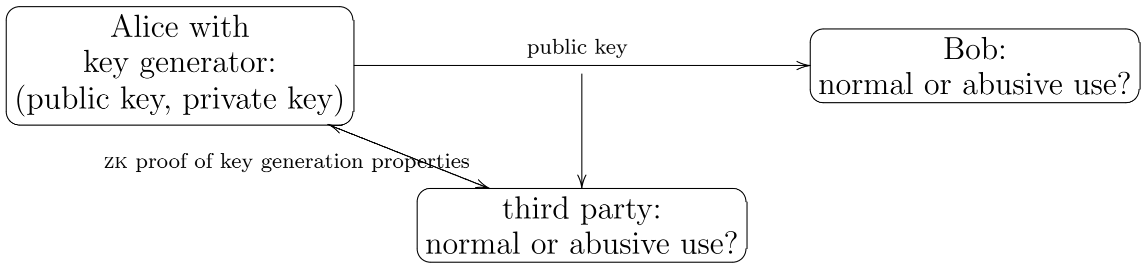

Alice with key generator: (public key, private key)
public key
Bob: normal or abusive use?
ZK proof of key generation properties
third party: normal or abusive use?

Figure 4–4: Desmedt's cryptosystem abuse via public key generation.

Another research area initiated by Simmon's work is the one of cryptosystem abuse. To abuse a cryptosystem is to use it in a way that it was not meant to. In 1988, Desmedt generalized the forms that cryptosystem abuses can take [Des88]. In particular, RSA public keys can be used to store additional information which the party that generated them wishes to transmit. For instance, Alice and Bob could appear, from the third party's point of view, to be sending messages with digital signatures. Meanwhile, they would additionally be secretly communicating using public keys that were generated to be of a special form which the other party can detect. Desmedt's solution to counter this abuse consists in a zero-knowledge (ZK) proof that the keys were generated according to some properties.

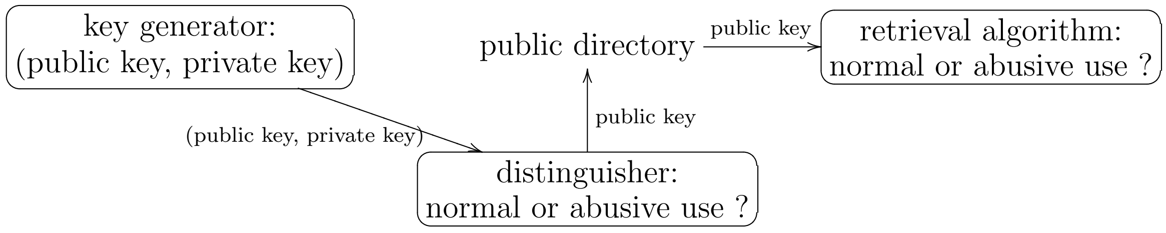


Figure 4–5: Desmedt's abuse via public key generation where the message is a backdoor. Compare with Figure 4–4.

A backdoor is a special type of cryptosystem abuse where public keys are generated as to give the backdoor's designer access to the corresponding private keys. Refer to Table 4–1 for the correspondences of the involved parties. Hence, to trans-

General abuse: Figure 4–4	Backdoored keys: Figure 4–5
Alice	backdoored key generator
Bob	designer (its retrieval algorithm)
third party	distinguisher

Table 4–1: Correspondences of parties involved in cryptosystem abuse and backdoor.

mit a backdoor to the third party's public key, the communication is not direct from Alice to Bob - from the key generator to the retrieval algorithm. Instead, it goes through the acceptance of the warden - the distinguisher - who publishes the message as his public key on a public directory.

Desmedt's 1988 paper included a solution, ZK, to the problem posed. However, theoretical results on backdoors that followed Desmedt's do not use ZK or refer to it. This work likewise does not allow the use of ZK as an interaction between the key generator and the party interested in having honestly generated keys. Therefore, the interaction is more limited.

On one hand, such an interaction does not use any additional protocol, making the setting more realistic. On the other hand, this restriction yields richer and more complex interactions between the parties involved with backdoors. In other words, because the problem is not assumed or given to be solved by the - perhaps - overkill of ZK, the study of backdoors is interesting.

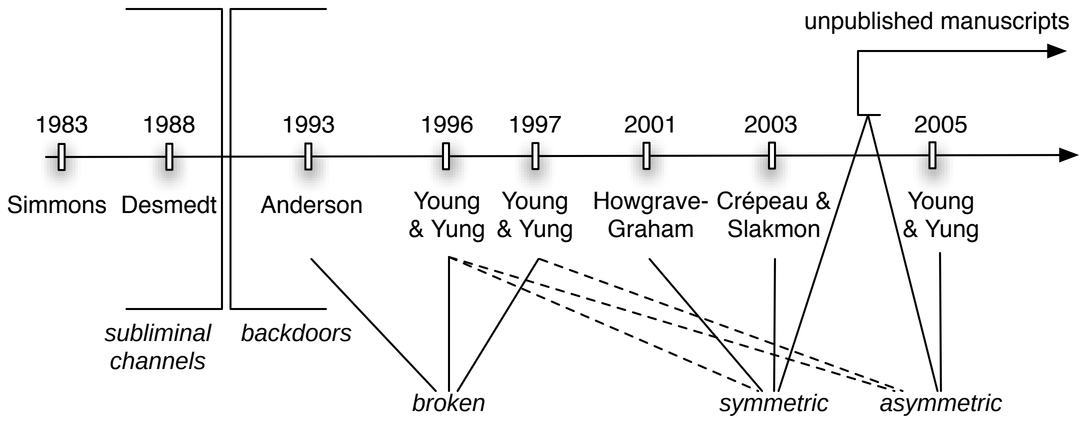


Figure 4–6: Time line of the discovery of backdoors in the key generation of common public-key cryptosystems. The *unpublished manuscripts* are from this work's author (some, collaboratively). The qualities of *symmetric* and *asymmetric* for backdoors correspond to the ones for cryptosystems; details are given in the next two sections.

There are two main streams of theoretical results on backdoored key generation. *Kleptography* consists of results from Young and Yung [YY96, YY97a, YY97b, YY05a, YY05b] (Section 6.2). The second stream consists of results from Crépeau et al. [CS03, ACS, ACK] (Section 6.3 and Chapter 7). Other backdoor results include the ones of Anderson [And93], broken by Kaliski [Kal93], and Howgrave-Graham [HG01] (Section 6.1). Figure 4–6 illustrates this as a timeline.

4.2 Definitions

4.2.1 Informal definition of a backdoor

One goal is to work out how much power a backdoored key generator may have. Recall the following vocabulary. The designer of a backdoored key generator is referred to as the *designer*. The user of a backdoored key generator is referred to as the *distinguisher*.

A backdoored key generator is a special type of malicious key generator. Three ideas capture the essence of the design of a backdoored key generator.

1. Let G_0 be an honest key generation algorithm that outputs legitimate keys as if sampled uniformly at random from the key space KS. For RSA, G_0 is as given in Subsection 2.2.4. Let k be the security parameter of the cryptosystem that includes G_0. Consider G_0 as a function that takes 1^k to key pairs. Therefore,

$$(k_{priv}, k_{pub}) = G_0(1^k) \quad \Longleftrightarrow \quad (k_{priv}, k_{pub}) \in_U KS$$

where $\in_U KS$ means that the keys are chosen randomly and uniformly in KS.

Let G_1 be a malicious key generation algorithm that outputs backdoored keys from the key space KSM with a given distribution function. Therefore,

$$(k_{priv}, k_{pub}) = G_1(1^k) \quad \Longleftrightarrow \quad (k_{priv}, k_{pub}) \in_R KSM \subset KS$$

where $\in_R KSM$ means that the keys are chosen randomly in KSM according to some distribution function.

2. The underlying structure of KSM can be expressed by the existence of two functions: E_{KSM} and D_{KSM}. If G_1 generates the pair $(k_{priv}, k_{pub}) \in KSM$, the following properties hold:
   (a) $k_{pub} = E_{KSM}(k_{priv})$
   (b) $k_{priv} = D_{KSM}(k_{pub})$

   where the private key k_{priv} can be understood to stand for any piece of information that allows to compute the private key.

   The related functions [2] E_{KSM} and D_{KSM} can be seen as the encryption and decryption functions of the generator G_1, concealing and retrieving the backdoor information, respectively. This pair of functions can either be based on a symmetric or an asymmetric cryptosystem.
3. The backdoored key generator, G_1, should not be distinguishable for the honest generator, G_0.

[2] The fact that more than one k_{priv} may correspond to a k_{pub} is incorporated in this chapter's definitions with an additional step of randomization. Therefore, E and D may still be considered to be functions, for simplicity.

Let $\mathcal{O}_{G_0}$ and $\mathcal{O}_{G_1}$ be oracles that randomly sample KS and KSM, respectively. Suppose that the distinguisher requests a key, then it is received from one of $\{\mathcal{O}_{G_0}, \mathcal{O}_{G_1}\}$ uniformly at random. The goal of the distinguisher is to determine whether $\mathcal{O}_{G_0}$ or $\mathcal{O}_{G_1}$ has been used to generate the key pair. For this, the distinguisher may use information contained in the inputs and outputs of the generators, as well as information derived from external measures on the generator such as how much time it consumes. Cf. Figure 4–7.

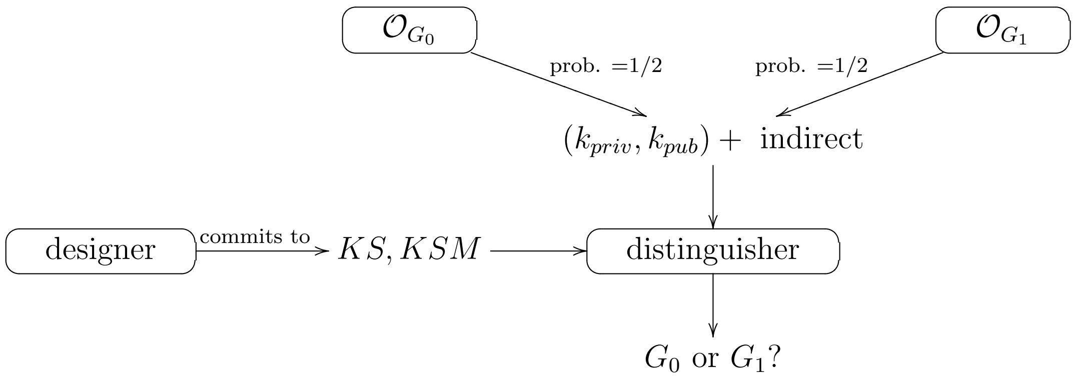


Figure 4–7: Third idea that captures the essence of the design of a backdoored key generator: indistinguishability. Schematics based on Figure 4–5.

In the literature, an hypothetical polynomial-time black box for an algorithm or function, A, is called an oracle for A, denoted by $\mathcal{O}_A$. This means that no function nor algorithm that computes A is explicitly given. However, in this work, the notion of oracle captures physical black box implementation of a given algorithm. Therefore, external measure can be taken during the oracle's processing of its input. Using physical oracles instead of black boxes allows one to use the standard notation and basic results on oracles. Therefore, this choice yields a better formalization for our theoretical approach to backdoors.

Finally, we will not develop an extensive notation for external measures, because results regarding them are straightforward. For instance, we are interested in computing the time complexity of backdoored key generators.

4.2.2 Model of analysis

The physical oracles used to implement the generators G_0 and G_1 can be analyzed according to two different paradigms.

Definition 4.2.1 *A* **black box** *analysis of a physical oracle assumes that the distinguisher has access to the inputs, the outputs and the external physical measures taken during the computations of the oracle.*

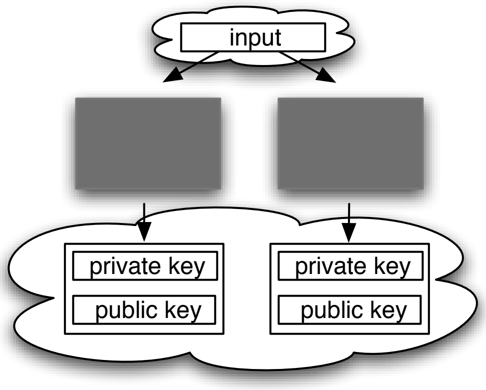


Figure 4–8: Two physical oracles outputting public key pairs. Which is G_0? Which is G_1?

However, in this work, a stronger distinguisher is considered. While G_0 is based on a published algorithm, what is known on G_1?

Definition 4.2.2 *A* **white box** *analysis of a physical oracle assumes that the distinguisher has access to the inputs, the outputs, the external physical measures taken during the computations of the oracle as well as the description of the underlying algorithm.*

Two important clarifications are in order. Firstly, in the case of Definition 4.2.2, the distinguisher *does not* have access to the internal trace of the execution. In particular, no access to internal implementation details such as variables is possible. This approach is similar to the analysis of probabilistic cryptographic algorithms, for which the random choices are unknown.

Secondly, if the underlying encryption algorithm E_{KSM} relies on a symmetric cryptosystem, the white box analysis is provided with the decryption of the encryption algorithm, but not the secret key used. This is coherent with the usual models of analysis of symmetric and asymmetric cryptography.

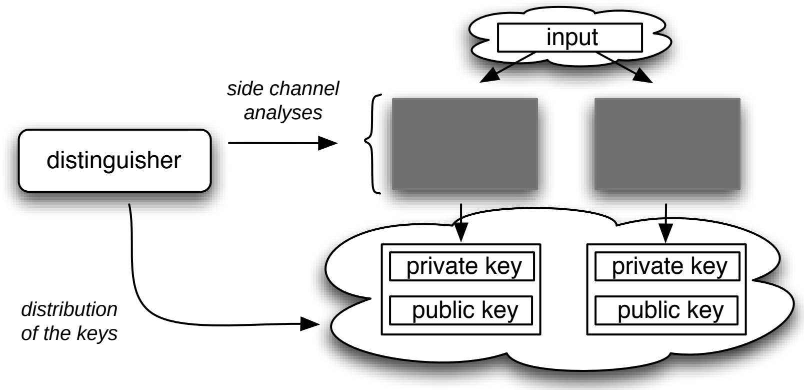


Figure 4–9: The distinguisher can deploy two main types of attacks against keys outputted by a physical oracle: side channel analyses and statistical analyses of the keys.

4.2.3 Formal definition of a backdoor

A more formal definition of a backdoored key generator is presented in this section. The E_{KSM} and D_{KSM} functions which capture the underlying structure of the malicious key space KSM of the backdoored key generator can be expressed as compositions of the following three functions and their inverses, respectively.

The first function's purpose is to extract from the private key, k_{priv}, the information to be concealed in the public key, k_{pub}. There are many possibilities for I, but they must satisfy the following definition.

Definition 4.2.3 (Information compression function) *Let I be a function of k_{priv} which output consists in sufficient information to efficiently retrieve k_{priv}.*

To be useful, I should be such that $|I(k_{priv})| < |k_{priv}|$ and $|I(k_{priv})| \ll |k_{pub}|$. The second property is essential in order to leave bits free for the randomization of I's output via the encryption function, E.

Example 4.2.4 For RSA keys, for instance, this information can be half the most significant bits of one of the prime factors of $n = pq$. Using Theorem B.1.1, one can factor the public modulus, n, (Subsection 2.2.7) and retrieve the private key, d. $\square$

The second function is this encryption function, E [3] . Aside from being invertible, E's only required property is that its output's distribution be computationally

[3] In this work, the usual multi-value probabilistic encryption is done in the following particular way. The encryption function, E, is taken as deterministic and the randomness is added separately. This allows to separate the accounting of the backdoored key set, which depends on this randomness.

indistinguishable from uniform. Knowledge on the distribution of E's output is essential to assess the distribution of the resulting backdoored public key. This requirement is straightforward to comply to in the symmetric E case. In the asymmetric E case, it is feasible in particular for RSA, via OAEP (end of Subsection 2.2.8), and for EG, via ECIES (Remark A.3.7).

Definition 4.2.5 (Encryption function) *Let E be a function of the information which is to be concealed $I(k_{priv})$. This function corresponds either to:*

1. *An invertible pseudo-random-like function, π_β, keyed with β, such that π_β's output distribution is computationally indistinguishable from uniform. This is the symmetric version of E.*
2. *A trapdoor one-way function E, based on a public-key cryptosystem, such that there are conditions that can be satisfied so that E's output distribution is computationally indistinguishable from uniform. To E corresponds an inverse function, D, which uses the trapdoor. This is the asymmetric version of E.*

This insures that the designer is the only party able to use the backdoor.

Example 4.2.6 Consider another example with RSA keys, from [YY96] (the algorithm is given in Figure 6–4). The prime $p = I(k_{priv})$ and it is encrypted using $E =$ RSA again, but with modulus N of the size of p.

The original paper does not mention the use of OAEP, but this is a minor modification that does not change the spirit of this algorithm. However, as noted in [CS03], that the encryption of the backdoor information be distributed close to uniformly may be undesirable (but this has more to do with the third function). □

The third function concerns the bits which the encryption of the backdoor information are set to occupy, in the backdoored public key. In other words, this function concerns the location of the embedding, and thus is called the *embedding function*.

Definition 4.2.7 (Embedding function) *Let f_R be a function of the encrypted information, $E \circ I(k_{priv})$. This function corresponds to:*

1. *the bit location of the insertion of $E \circ I(k_{priv})$,*
2. *the probabilistic assignation of values to the bits not involved in this location.*

and, with it, the public key is finalized as $k_{pub} = f_R \circ E \circ I(k_{priv})$, for R random.

To be useful, f_R should be invertible and insure that the encrypted information, $E \circ I(k_{priv})$, is placed in k_{pub} such that the backdoored keys look like honest keys (this is to be defined formally later, in Subsection 5.1.3).

Example 4.2.8 Consider the same algorithm with RSA keys as in the previous example, from [YY96]. The encrypted information $E(p)$ is embedded in e. All the bits of e are involved in this embedding, but the ones of n are entirely generated as in the honest algorithm. □

The resulting backdoored RSA public key is

$$(n, e) = f_R \circ E \circ I(d),$$

for R random.

With these definitions, a formal definition of a backdoored key generator can be presented.

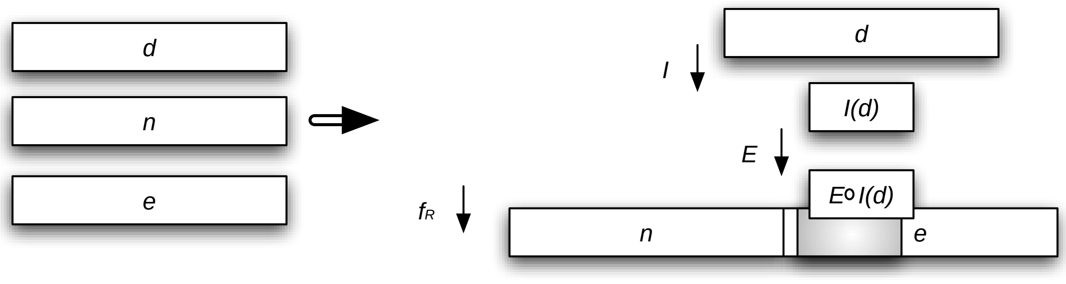


Figure 4–10: Left, the honest key parameters of the RSA cryptosystem: d is private and (n, e) is public. Right, an illustration of the functions involved in defining backdoored key generation. Function I extracts information from d. This information is encrypted via E. The encrypted information is well-distributed in (n, e) via f: the shaded part of (n, e) is where $E \circ I(d)$ is embedded by f.

Recall that the public-key cryptosystem's (honest) key generator is denoted by G_0, such that $G_0(1^k) = (k_{pub}, k_{priv})$. On input corresponding to the security parameter k, G_0 outputs a pair consisting of a public key, k_{pub}, and a private key, k_{priv}.

Definition 4.2.9 (Formal) *Given a public-key cryptosystem's key generator, G_0, a designer produces a key generator, G_1 and a retrieval algorithm R_1. Let I be the information compression function (Definition 4.2.3). Let E be the encryption function (Definition 4.2.5), with its inverse function D. Let f_R be the embedding function (Definition 4.2.7). The keys outputted by G_1 are (securely)* **backdoored** *if $G_1(1^k)$ computes (k_{pub}, k_{priv}) such that the following properties hold:*

1. **Confidentiality:** *$k_{pub} = f_R \circ E \circ I(k_{priv})$ is not computationally invertible.*
2. **Completeness:** *The functions I and f_R are invertible, so the retrieval algorithm, R_1, computes $k_{priv} = R_1(f_R \circ E \circ I(k_{priv}))$.*

3. **Indistinguishability:** *The designer commits to the key space $KSM \subseteq KS$, corresponding to the keyspaces of G_1 and G_0, respectively. Then*

(a) the outputs of oracles $\mathcal{O}_{G_0}$ and $\mathcal{O}_{G_1}$ are perfectly, statistically, or computationally indistinguishable;

(b) the external measures on the physical oracles $\mathcal{O}_{G_0}$ and $\mathcal{O}_{G_1}$ cannot significantly distinguish one from the other.

Refer to Figure 4–11 for an illustration of this definition.

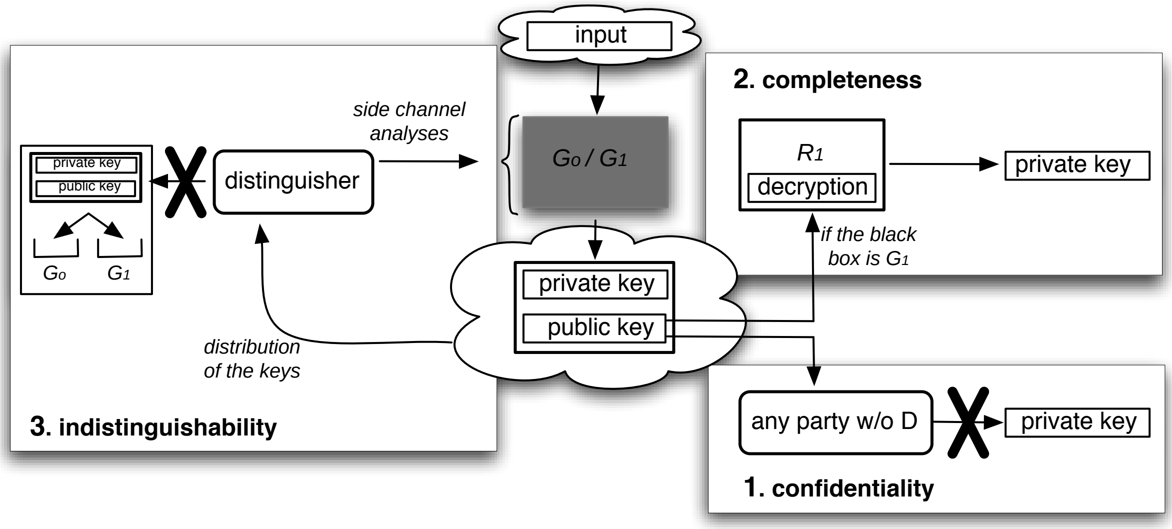


Figure 4–11: More formal definition of a backdoored key generator G_1. The three key conditions are framed. Compare with Figure 4–9.

Confidentiality means that it is computationally infeasible to retrieve k_{priv} from k_{pub}. Completeness means that the designer may compute k_{priv} from k_{pub}. The spirit of these definitions is taken from Young and Yung [YY05a]. However, they define a weaker key indistinguishability as the keys being computationally indistinguishable (which is the weakest variant), and without external measures.

Assumption on E. Because the length of the generated keys is determined by the distinguisher (by setting the generator input to 1^k), the cryptosystem involving E, denoted by $\mathcal{C}_E$, should be at least as strong as the one involving G_0, denoted by $\mathcal{C}_{G_0}$. Otherwise, the backdoored key generator may be trivially insecure.

Suppose that it is not the case and suppose that the distinguisher chooses k of minimal length such that $\mathcal{C}_{G_0}$ is secure for a time T. Because it is weaker, $\mathcal{C}_E$ may be expectedly secure only for a shorter period of time $t \ll T$.

Example 4.2.10 Consider the same algorithm with RSA keys as in the previous example, from [YY96]. In this algorithm, $\mathcal{C}_E$ uses a key length of half the one of $\mathcal{C}_{G_0}$. For instance, RSA-1024 is believed to be secure for 600 months, but this backdoor information is encrypted with RSA-512, which is breakable in less than 5 minutes (cf. Table A.1).

Therefore, after such a short time, the confidentiality of the backdoored key generator fails, which allows any party to retrieve the legitimate user's private key. Thus the keys are distinguishable and the backdoor is "opened" to any party. $\square$

Minimality of Definition 4.2.9. Can a simpler and equivalent variant of the preceding definition be formulated? It may seem that a simpler definition could discard Point 1 because confidentiality is a sub-feature of indistinguishability: if the keys are indistinguishable for the designer, then this party may not, in particular, find its own private key embedded in its public key. Nevertheless, in some cases, confidentiality may hold while indistinguishability fails. Therefore, taking apart confidentiality from indistinguishability allows a more precise analysis of different backdoored key generators. Consequently, Definition 4.2.9 does not call for trimming.

Injectivity of $f_R \circ E \circ I(k_{priv})$. In order for $I^{-1} \circ D \circ f_R^{-1}(k_{pub})$ to exist, the function $f_R \circ E \circ I(k_{priv})$ must be injective. In some of the backdoored key generators that are examined in the following chapters, $f_R \circ E \circ I(k_{priv})$, or more precisely, the $f_R \circ E$ component, is not injective. However, approximate injectivity is then showed, which means that, in such algorithms, there is a collision only negligibly often, when retrieving the private key from a backdoored public key.

4.3 Comparison with SETUP definitions for backdoors in cryptosystems

One way of constructing backdoors can be traced back to five papers on kleptography by Young and Yung [YY96, YY97a, YY97b, YY05a, YY05b]. Kleptography is the study of backdoors for cryptosystems, i.e., this field of study is not limited to key generators as this work is. In kleptography, a backdoor is called a SETUP, which is an acronym for Secretly Embedded Trapdoor with Universal Protection.

4.3.1 Definition of SETUP

The following Table 4–2 summarized the different versions of the definition of a SETUP. The properties of completeness, confidentiality and indistinguishability are as defined in Definition 4.2.9. The white box effect states whether or not, upon discovery of the description of C, the backdoored keys remain non-determinable (only the designer can generate past or future backdoored keys) and indistinguishable.

Definition 4.3.1 *Let C_0 be the honest cryptosystem with input and output specifications. A* **setup cryptosystem** *is denoted by C and conforms to these specifications and satisfies the following properties:*

SETUP		completeness	confidentiality	indistinguishability	white box effect
[YY96]		yes	yes	everyone	n/a
[YY97a], [YY97b]	Def. 1	yes	yes	everyone	non-determinable
	Def. 2	yes	yes	everyone, except the distinguisher	non-determinable
	Def. 3	yes	yes	everyone	non-determinable and indistinguishable
[YY05a], [YY05b]		except with negligible probability	except with negligible probability	everyone, except with negligible probability	non-determinable and indistinguishable

Table 4–2: Evolution of definitions in kleptography with respect to the more abstract notions of confidentiality, completeness, indistinguishability, and white box effect.

4.3.2 Evolution of algorithms w.r.t. definitions in kleptography

The following table keeps track of the algorithms provided to satisfy the SETUP definitions. For backdoored key generation SETUPs, the two last columns indicate whether the definitions of Table 4–2 were attained. Overall, the definitions are satisfied only for EG key generation in [YY96].

SETUP publication		type	asymmetry	security
[YY96]		RSA key generation	yes	based on RSA, but broken: bad key lengths
		EG key generation	yes	based on RSA or EG, but small number of keys
[YY97a]	regular	n.a.	n.a.	n.a.
	weak	n.a.	n.a.	n.a.
	strong	DH key exchange	n.a.	n.a.
		RSA key generation	yes	based on DH, but broken: bad key lengths
[YY97b]	regular	discrete log problem, EG signature	n.a.	n.a.
	weak	n.a.	n.a.	n.a.
	strong	EG encryption	n.a.	n.a.
[YY05a]		RSA key generation	no (var. i)	based on Rabin, but broken: bad key lengths
[YY05b]		RSA key generation	yes	based on elliptic curve DH, but complicated

Table 4–3: Evolution of algorithms w.r.t. definitions in kleptography.

4.3.3 White box effect, uniformity and asymmetry

To simplify the white box effect and other reverse-engineering considerations, Young and Yung introduced the notion of uniformity.

Definition 4.3.2 Uniformity *is said to hold if the* SETUP *attack is the same in every instance of* C.

Uniformity means that the backdoored key generator is the same for all the distinguishers for whom it generates keys. Therefore, there is no secret portion in the designer's encryption function, not even keys. If uniformity holds, then the backdoored key generator's code can be revealed (this is the white box effect) without affecting indistinguishability. Furthermore, if a backdoored key generator is asymmetric, then uniformity holds, because revealing an asymmetric key does not provide additional information, computationally speaking.

However, there are no known secure examples of RSA SETUPs which were purely asymmetric, as illustrated in Table 4–3. There are some asymmetric SETUPs examples for EG, but only with a small number of generable keys [YY96], given in Figures 6–6, 6–7 and 6–8.

4.3.4 Asymmetry (un)satisfied by SETUPs

The SETUP definitions are such that the designer's cryptosystem is asymmetric. However, this does not necessarily imply that the SETUPs provided to correspond to the definitions are asymmetric backdoors. In the RSA backdoored key generations of [YY96, YY97a, YY97b], the key lengths are not secure (Definition 4.2.9). In other cases, some symmetric cryptography is also used. Only one of these cases concerns key generation, in [YY05a], which algorithm is given in Figure 6–11.

Uniformity holds for the backdoored key generator of [YY05b] as this SETUP is asymmetric, as claimed. At first, it seems that [YY05b] uses a private parameter, s_{priv}, to generate the prime p, as shown in the algorithms of Figure 6–12. Confidentiality appears to fail if s_{priv} is not kept secret, so it appears that s_{priv} cannot be the same for all the distinguishers. Nevertheless, uniformity does hold and this backdoor is indeed asymmetric, because s_{priv} is used prior to the backdoor mechanics themselves.

This algorithm is analyzed in details in Subsection 6.2.4.

4.4 Lexicon

- **backdoor**: means by which a cryptosystems private parameters can be recovered, using separate information; in practical situations, this is often understood as the information consisting of the *designer*'s key; in theory, thus in this work, *backdoored keys* are outputted by a *contaminated* algorithm, so the means of recovery consist in both this special algorithm and the designer's key
- **backdoored key** (Definition 4.2.9): key from a *contaminated* key generator, that is, the backdoored key generator, G_1, of Subsection 4.2.1
- **black box**: object conceived in terms of its input and output, i.e. its internal functioning is abstracted (Figure 4–8)
- **contamination (of an algorithm)**: the embedding of a *backdoor* in an algorithm by an *adversary*, called a *designer*

- **designer**: the adversarial party that aims at recovering protected private information, e.g. other parties' private keys (Figure 4–3), via a backdoor
- **distinguisher**: the *legitimate user*, when it is an *adversary* against the *designer*
- **generable**: said of a key which can be generated by a given algorithm
- **honest key**: key produced by the standard key generation algorithm, which has not been subject to *contamination*, that is, the backdoored key generator, G_0, of Subsection 4.2.1
- **key escrow**: decryption keys are held in escrow by a third party, so that some party can retrieve them (Figure 4–1)
- **legitimate user (of the cryptosystem's public key)**: the sole party who should know the corresponding secret key; takes the role of the *adversary* for the distinguishability of the honest and backdoored keys (Figures 4–2 and 4–3)
- **white box**: object which description is known

4.5 Chapter notes

Subsection 4.1.1. Background: Two contexts. A reference for key escrow is [MvV01, Section 13.8.3]. The etymology for escrow is taken from the *American Oxford Dictionaries*, online.

Subsection 4.1.3. Background: History. One research area that Simmons' work initiated is the one of information hiding, or steganography. A general reference on this area is [KP99].

CHAPTER 5
Measures for backdoored keys

The aim of this chapter is to allow a classification of different approaches used to analyze backdoored key generators. For this purpose, several metrics are introduced to create different classes of comparison.

Ultimately, perfect malicious key generators would be indistinguishable from honest key generators, in terms of number of keys, key distribution, side channel information, and remain so upon complete reverse-engineering of the generators. The formal goal is to measure how an algorithm satisfies both parts of Property 3 of Definition 4.2.9.

- indistinguishability
  - nature of the keys (static)
    - cardinality
    - probability distribution
      - distribution properties
      - variable correlations
    - symmetry / asymmetry
  - side channels (dynamic)
    - complexity
    - memory

5.1 Nature of the keys

One important issue regarding the quality of backdoored key generators is the indistinguishability of backdoored keys from honest ones, with respect to their nature. Obviously, this issue is often addressed, even if only implicitly. The formal goal is to measure how an algorithm satisfies Property 3(a) of Definition 4.2.9.

5.1.1 Classical definitions of indistinguishability

Indistinguishability is defined for ensembles. Consider sets of elements of size ℓ, with their corresponding probability distributions. Then consider a family of such sets parametrized with ℓ. Such a probability distribution on $\{0,1\}^\ell$ is called an ensemble, and is usually denoted $\mathcal{D}_\ell$ [Lub96, p.6]. Ensembles can be indistinguishable in commonly defined three ways:

1. *computational indistinguishable*: if there exists no polynomial-time algorithm that can significantly distinguish them;
2. *statistical indistinguishable*: if the distance between the two distributions is not significant;
3. *perfect indistinguishable*: if the two sets are the same.

A similar approach has been used in zero-knowledge (ZK). More formally, Young and Yung provide the following definition, where ℓ is the security parameter. Therefore the elements of KS are of size (in general, linear in) ℓ.

Definition 5.1.1 (Computational indistinguishability [YY05b]) *The distinguisher is given two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$. One of the oracles generates keys uniformly distributed in KS and the other one, with some given distribution in $KSM \subset KS$. If the distinguisher cannot distinguish in polynomial time (with respect to ℓ) which of*

$\mathcal{O}_0$ or $\mathcal{O}_1$ generates KS with probability significantly greater than $1/2$, then $\mathcal{O}_0$ and $\mathcal{O}_1$ are **computationally indistinguishable**.

Suppose that we have extracted all the information about the distributions of both KS and KSM. Suppose that KS has distribution D_{G_0} and KSM has distribution D_{G_1}. If $i \in KS$ is not defined in $\mathcal{D}_{G_1}$, its probability is zero for that distribution.

Definition 5.1.2 (Statistical indistinguishability [Lub96], p.70) *The distribution D_{G_1} on the set KSM has a statistical distance of ε w.r.t. D_{G_0} on KS if*

$$\max_{T \subseteq S} \left\{ \sum_{i \in T} |h_i - g_i| \right\} = \varepsilon$$

where the probability of $i \in KS$ according to D_{G_0} is h_i and according to D_{G_1} is g_i.

Pros and cons of classical definitions of indistinguishability

Pros of statistical indistinguishability. Definition 5.1.2 would arguably produce the strongest possible statements. The perfect case would be for all keys to contain a backdoor, which is when $\varepsilon = 0$. This is not a useful consideration because the cryptosystem would be considered broken, which normally is unknown or infeasible. Therefore, values of ε yield a family of distributions associated with KSM sets. As ε becomes more negligible w.r.t. ℓ, these distribution become closer to the perfect case.

Moreover, Definition 5.1.2 provides an absolute measure which is stronger than a definition based on computational indistinguishability. It takes into account features of the ensemble only, because it only depends on distributions. Therefore, this measure is independent of computational issues. One reason in favor of computational indistinguishability is that it is the minimum theoretical security requirement so that the distinguisher satisfies Property 3(a) of Definition 4.2.9. However, no rigorous proof of such a definition being satisfied exists, despite a few number of attempts. The most recent one is detailed and criticized in Subsection 6.2.4.

Cons of statistical indistinguishability. However, obtaining information on the distributions of honest or backdoored keys is a considerable issue. For instance, for the honest RSA key generation, the analysis of the distribution of the product of two primes of the same length is non-trivial. Thus, for RSA backdoors, the honest distribution, to be used for comparison in analyzing backdoored distributions, is difficult to establish.

Middle ground. Thus a more practical indistinguishability definition than statistical indistinguishability is sought. The more approximate notion of *diversity*, which this work defines, is set in between computational indistinguishability and statistical indistinguishability. A first reason for this new definition is that there exists no proof of computational indistinguishability so far: the one in [YY05b, Appendix A.1] is incomplete, as shown at the end of Section 6.2. The second reason is that our definition of diversity allows the analysis of algorithms that approach the ultimate property of indistinguishability. Diversity is useful in Chapters 6 and 7, in order to compare the characteristics of various backdoored key generators.

Diversity relates to the cardinality of the key spaces as well as to the key distributions. The intuition is that both concepts of statistical and computational indistinguishability could be combined, as to provide as much useful information as possible. There are some absolute quantities that can be deduced about key distribution, such as the approximate total number of generable keys. However, G_1 usually relies on a form of pseudo-randomness used to produce a distribution that approaches the uniform one. If it is the case, computational statements are unavoidable, unless one completely analyzes the distributions produced by the involved pseudo-random function(s).

5.1.2 Cardinality

The first step in defining the notion of key diversity is the notion of cardinality. This is the cardinality of the key space KSM w.r.t. the one of the key space KS. As an example, consider Figure 5–1, illustrating the case of k_{pub} from Definition 4.2.9 being an RSA public key and the designer's encryption function E is the Elliptic Curve Integrated Encryption Scheme (ECIES) (Appendix A.3). The security parameter of a cryptosystem $\mathcal{C}$ is denoted by $\ell_{\mathcal{C}}$. Obviously, the value of $\ell_{\mathcal{C}}$ varies with time: as cryptanalysis evolves, longer keys can be efficiently broken, and these lengths yield lower bounds on $\ell_{\mathcal{C}}$. This accordingly increases the value of $\ell_{\mathcal{C}}$.

Let G_0 be a key generator. If necessary for disambiguation, instead of k_{pub}, let $k_{p(G_0)}$ denote the public parameter of G_0: for example, $k_{p(\mathrm{RSA})} = (n, e)$. It can be assumed that the length of $k_{p(G_0)}$, $|k_{p(G_0)}| = \ell_{p(G_0)}$, is minimal in order to achieve security (honest) key generation.

Let G_1 be a backdoored key generator that mimics G_0. Suppose that we use algorithm E to encrypt the backdoor into k_B, which is a part or all of $k_{p(G_1)}$. Algorithm E is such that $E : \{0,1\}^{n(\ell_E)} \rightarrow \{0,1\}^{m(\ell_E)}$ where ℓ_E is generally the length for which security is considered to be achieved (for the backdoor).

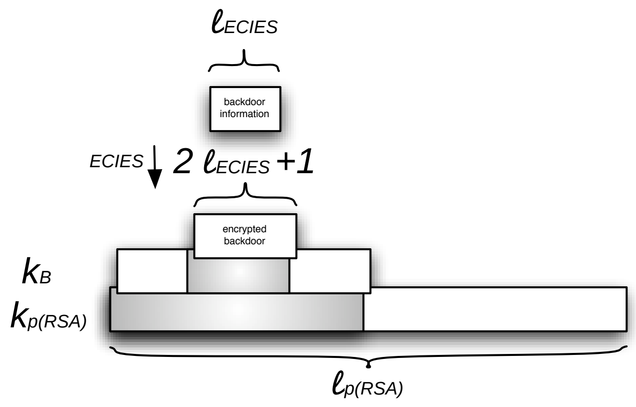


Figure 5–1: Backdoored RSA key encrypted via ECIES. The encrypted backdoor information is embedded in part of the public key, k_B. The shaded part of k_B is where the encryption of the backdoor is embedded. The shaded part of the public key $k_{p(RSA)}$ is where k_B is embedded (it is so, whether it contains a backdoor or not).

Note that k_B is not a subset of bits, it is a component (a variable) which is a part of the public key. For RSA backdoored key generators, k_B is either n or e.

Example 5.1.3 Consider again the backdoor of Example 4.2.6 (the algorithm is given in Figure 6–4). Recall that the prime $p = I(k_{priv})$ is encrypted by using $E =$ RSA again, but with modulus N of the size of p. This encrypted value is stored in e.

In the notation introduced, $k_B = e$ and $G_0 = E =$ RSA. However, $n(\ell_E) = m(\ell_E) = 2\ell_E$ and $\ell_E = k$, but this backdoored generator uses $k/2$ instead of the proper ℓ_E. Example 4.2.10 explains how the security of the algorithm is failed by this ℓ_E issue. □

Example 5.1.4 The backdoored generator illustrated in the preceding Figure 5–1 encrypts the backdoor information with ECIES and embeds it in RSA keys (the algorithm is given in Figure 7–15). From a result of Coppersmith (Corollary 6.1.8), the knowledge of some $k/2$ bits is sufficient to reproduce a prime p of k bits. The backdoored generator encrypts these $k/2$ bits with ECIES, yielding a ciphertext of $k+1$ bits. This ciphertext is finally embedded in n.

In the notation introduced, $k_B = n$, $G_0 = $ RSA, $E = $ ECIES, $n(\ell_E) = \ell_E$, $m(\ell_E) = 2\ell_E + 1$ and $\ell_E = $ length of the public prime used in ECIES. $\square$

Total number of random bits. The cardinality of KSM is proportional to the number of random bits used to generate $k_{p(G_1)}$. From Definition 4.2.9, $k_{p(G_1)}$ is generated as

$$k_{p(G_1)} = \underbrace{f_R}_{\substack{\text{embeds } E \circ I(k_{priv}) \\ \text{and random } R\text{:} \\ |R| \text{ random bits}}} \circ \underbrace{\underbrace{E}_{\substack{\text{probabilistic encryption} \\ \text{or trapdoor} \\ \text{one-way permutation:} \\ r(E) \text{ random bits}}} \circ \underbrace{I(k_{priv})}_{\substack{\text{fixed for } (I, k_{priv})\text{:} \\ i(k_{priv}) \text{ bits of} \\ \text{information}}}}_{\text{encrypted backdoor embedded in } k_B} .$$

The number of backdoored keys generable by G_1 is given by:

$$\begin{aligned} \mathcal{N}_{G_1} &= \mathcal{N}_{f_R} + \mathcal{N}_E + \mathcal{N}_{I(k_{priv})} \\ &= 2^{|R|+r(E)+i(k_{priv})} \cdot r_{\ell_{p(G_0)}}, \end{aligned} \tag{5.1}$$

where the second factor in Equation 5.1 accounts for the fact that there are $\ell_{p(G_0)}$-bit strings that are not valid $k_{p(G_0)}$ values. In other words, $r_{\ell_{p(G_0)}}$ is the ratio of accepted $\ell_{p(G_0)}$-bit strings. In general,

$$r_{\ell_{p(G_0)}} = \frac{KS}{2^{\ell_{p(G_0)}}}$$

so it is dependent on G_0 only. In other words, it is the fraction of keys that are accepted by G_0. As G_1 uses the same rejection criteria as G_0, this ratio affects the number of backdoored keys generable by G_1.

Location of the embedding. In no cases will the embedding be done in many parts of a public key. For instance, for RSA, no embedding may split the backdoor information between n and e. While it may appear to be beneficial to spread the information, this would readily forbid the generalized key regeneration that we will elaborate on in Subsection 5.1.4.

Consequently, it is not useful to account the number of generable key parts that do not participate in embedding the backdoor information. For instance, for RSA, if information corresponding to p is embedded in n, then the number of random bits used to generate the e part of the public key are discarded. Similarly, if the information corresponding to d is embedded in e, then the number of generable n is discarded.

These are cases where it is useful to compute the number of generable keys for a fixed parameter. For instance, in RSA, if one wishes to compute the number of backdoored e for a fixed n, one denotes this number by $\mathcal{N}_{G_1,e}$. That the two quantities,

$$\mathcal{N}_{G_1,n} \text{ and } \mathcal{N}_{G_1,e},$$

are denoted similarily is however an abuse of notation, because e is dependent on n. More formally, $\mathcal{N}_{G_1,e}$ means "$\mathcal{N}_{G_1,e}$ given n".

Example 5.1.5 Consider the PAP example from [YY96] (the algorithm is given in Figure 6–5). A backdoor is embedded in RSA keys such that p is hidden in n, i.e.

$I(k_{priv}) = p$ and $k_B = n$. More precisely, the prime p is scrambled as ρ via a fixed pseudo-random function [1] . The embedding function sets $n = \rho : r$, for a k-bit random string r. The other prime is set as $q = (\rho : r)/p$. Subsection 7.4.1 shows an analysis of a generalized version of this backdoor embedding strategy.

Firstly, compute how likely it is that the produced q is an integer and then how likely that it is also prime. The probability that p divides $\rho{:}r$ is given by the fraction of strings covered by multiples of p times the expected fraction of strings covered by $\rho{:}r$. This is:

$$\begin{aligned}\Pr\left[\frac{\rho{:}r}{p} \in \mathbb{Z}\right] &= \frac{\#\{2k\text{-bit multiples of } p\}}{2^{2k}} \cdot \frac{\#\{\rho{:}r\}}{2^{2k}} \\ &= \frac{2^k}{2^{2k}} \cdot \frac{2^k}{2^{2k}} = \frac{1}{2^{2k}}.\end{aligned}$$

Nevertheless, substituting r by an appropriate r' allows for the exact division by p.

$$\Pr\left[\frac{\rho{:}r'}{p} \in \mathbb{Z}\right] = 1$$

Recall that the other (candidate) prime is set as $q = (\rho : r)/p$. In fact, the random choosing of r is not a useful operation: only two values of r are such that $\rho{:}r$ is a multiple of p, because $|r| = |p|$. If p is not a factor of $\rho{:}r$, then $\rho{:}r$ plus (or minus) a

[1] The precise statement in [YY96, p.5] is “to achieve a pseudo-randomness of the values” of ρ. This is meant in the sense of achieving a good distribution via a pseudo-random function F_K used as $\rho = F_K(p)$. The achieved distribution may be argued to be good, but it is not a theoretically pseudo-random one, since the key, K, is fixed.

remainder is a multiple of p. Therefore r' is r plus (or minus, resp.) this remainder. Since the resulting multiples are successive, one will be even and the other odd, thus only the latter can be prime. By Theorem 2.2.4, a k-bit integer, q, is prime with probability of about $2^{-\lg k}$. Therefore

$$\begin{aligned}\Pr\left[\frac{\rho\!:\!r'}{p} \text{ is prime}\right] &= \Pr\left[\frac{\rho\!:\!r'}{p} \in \mathbb{Z}\right] \cdot \Pr[q \text{ is prime } | q \in \mathbb{Z}] \\ &= 1 \cdot 2^{-\lg k} = \frac{1}{2^{\lg k}}.\end{aligned}$$

Secondly, sum up for all values of p to obtain the expected number of n. The number of p was approximated in Example 2.2.7. One expects

$$\begin{aligned}\mathcal{N}_{G_1,n} &= \#\{\rho\!:\!r' \mid \exists \text{ prime } q \text{ s.t. } pq = \rho\!:\!r'\} \\ &= \#\{p\} \cdot \#\{r'\} \cdot \Pr\left[\frac{\rho\!:\!r'}{p} \text{ is prime}\right] \\ &\approx 2^{k-\lg k} \cdot 1 \cdot 2^{-\lg k} \\ &= 2^{k-2\lg k}\end{aligned}$$

backdoored keys.

Finally, reformulate this expected number using the notation introduced in Equation 5.1. One obtains

$$\begin{aligned}r_{\ell_{p(G_0)}} &= \frac{KS}{2^{\ell_{p(G_0)}}} \\ &= \frac{(2^{k-\lg k})^2}{2^n} = 2^{-2\lg k}\end{aligned}$$

because the number of honest $n = pq$ is the square of the number of k-bit prime (given by Example 2.2.7) and the length of the portion of the public key which is the site of the embedding, n, is $\ell_{p(G_0)} = 2k$.

It is left to consider the number of bits in the backdoored keys. The number of bits of p is $i(k_{priv}) = k$. There are no random bits involved in its encryption, so $r(E) = 0$. It could at first appear that $|R| = |r| = k$. Nevertheless, the restriction that the subsequently computed q be prime makes the expected value of additional random bits from r is $|R| = 0$. Therefore

$$2^{|R|+r(E)+i(k_{priv})} \quad = \quad 2^{i(k_{priv})} = 2^k$$

and $\mathcal{N}_{G_1,n} = 2^{i(k_{priv})}\, r_{\ell_{p(G_0)}} = 2^{k-2\lg k}$ as before. $\square$

Example 5.1.6 Consider the same backdoored key generator as in Example 5.1.4, with $G_0 = $ RSA and $E = $ ECIES. Recall $k_{p(\text{RSA})} = (n, e)$, and take $k_B = n$, so from Section A.1, $\ell_B = 1024$. Therefore, $2k = |k_B| = 1024$. For ECIES, from Remark A.3.7, $n(\ell_E) = \ell_E$, $m(\ell_E) = 2\ell_E + 1$ and also from Section A.1, $\ell_E = 160$. More precisely, the $k/2$ MSB of p are pseudo-randomly generated from 160 bits denoted (x, K). Thus their distribution is computationally indistinguishable from uniform.

$$i(k_{priv}) \quad = \quad |x| + |K| = 160$$

In (x, K), the parameter K is typically the key and x, the seed of the pseudo-random generator. For instance, the generator can be instantiated with AES.

Denote the $k = 512$ LSB leftover bits of n by r_2. Denote the $k - m(\ell_E) = 512 - (2 \cdot 160 + 1) = 191$ MSB leftover bits of n by r_1.

$$n = r_1 : E(x, K) : r_2$$

At first, it would seem that the number of free bits is exactly the number of bits not needed to embed the backdoor. That is $|R| = |r_1| + |r_2|$. If it was the case, then the following would hold.

$$|R| \;\;=\;\; |k_B| - m(\ell_E) = 1024 - 2 \cdot 160 = 704.$$

However, the k LSB bits of n are as in Example 5.1.5, that is, these bits are not free. More precisely, when $q = \lfloor n/p \rfloor$ is set, the initial value of r_2 is substituted by a r_2' that allow integer division. This fixes the k LSB bits. Therefore

$$|R| = |r_1| = 191.$$

The number of random bits involved in encrypting (x, y) is, by Example A.3.5,

$$r(E) \;\;\approx\;\; 160.$$

Therefore

$$|R| + r(E) + i(k_{priv}) \approx 191 + 160 + 160 = 511.$$

From Example 2.2.6, the probability that a random k-bit integer is prime is

$$\frac{\pi(2^k)}{2^k} \approx 2^{-\lg k} = 2^{-9}.$$

Therefore,

$$\begin{aligned}\mathcal{N}_{G_1,n} &\approx 2^{|R|+r(E)+i(k_{priv})-2\lg k} \\ &\approx 2^{511-2\cdot 9} = 2^{493}.\end{aligned}$$

So the size of the backdoored set $\{(n,d,e)\}$ with fixed e is approximately 2^{493}. The number of keys generated by the standard RSA key generation, as in Subsection 2.2.4, with $\ell_{p(G_0)}/2 = 2k = 1024$, resolves to 2^{493}, from Example 2.2.9. This indicates that the cardinality of KSM is approximately about half of the total number of keys.

In both number of keys computations, there is a factor of $2^{-2\lg k}$ which comes from the number of invalid n that are not the product of two k-bit primes. Without this, the number of honest keys, with fixed e, is about $2^{2k} = 2^{1024}$. Also without discarding these invalid values of $2k$, the cardinality of KSM is about

$$2^{(512-(2\cdot 160+1))+160+160} = 2^{511},$$

where $m(\ell_E) = 2\cdot 160+1$ is the amount of space taken by the ECIES encryption, which is compensated by the amount of information, $n(\ell_E) = 160$ bits, and of randomness, $|K| \approx 160$ bits, within it. Varying the length of the ECIES security parameter, ℓ_E, does not significantly affect this compensation.

Because the used E is an encryption function, this process is injective, so E maps any two different values of information and randomness to a different image. Therefore, the number of images adds up in the same manner as the number of points in the domain of E. $\square$

Definition 5.1.7 (Number of keys) *Denote the* **number of honest keys generable by Algorithm** G_0 *by* $\mathcal{N}_{G_0}(\ell_{p(G_0)})$. *Denote the* **number of backdoored keys generable by Algorithm** G_1 *by* $\mathcal{N}_{G_1}(\ell_{p(G_0)}, \ell_E)$. *Denote the* **key ratio**, *which is the inverse of the number of times that there are more honest than backdoored keys, by* $\mathcal{R}_{G_1}(\ell_{p(G_0)}, \ell_E)$. *Then*

$$\begin{aligned}\mathcal{N}_{G_1}(\ell_{p(G_0)}, \ell_E) &= 2^{|R|+r(E)+i(k_{priv})}\, r_{\ell_{p(G_0)}},\\ \mathcal{R}_{G_1}(\ell_{p(G_0)}, \ell_E) &= \frac{\mathcal{N}_{G_1}(\ell_{p(G_0)}, \ell_E)}{\mathcal{N}_{G_0}(\ell_{p(G_0)})}.\end{aligned}$$

Example 5.1.8 In Example 5.1.6,

$$\begin{aligned}\mathcal{N}_{G_1}(\ell_{p(G_0)}, \ell_E) &\approx 2^{1024+493} = 2^{1517},\\ \mathcal{R}_{G_1}(\ell_{p(G_0)}, \ell_E) &\approx 2^{1517-2048} = 2^{-531}.\end{aligned}$$

□

Cardinality classification for RSA backdoored key generators. Follow the relative ratings for RSA backdoored key generators, in increasing order of desirability. The security parameter of G_0 and G_1 is k. In practice, the ratio of the cardinality of G_1 to the one of G_0 is of the form

$$2^{c \cdot k} \cdot f(k)$$

where $c \leq 0$ and $f(k)^{-1} \in 2^{o(k)}$. This is another way of expressing that $\lg(f(k)^{-1}) \in o(k)$. Informally, this means that $f(k)^{-1}$ is negligible w.r.t. 2^k, so that all the exponential factors are collected in the constant c.

Then, classes of values of c are meaningful. That $c = -1/2$ means that about $k/2$ bits of freedom in the key selection are lost because of their backdoored generation. The case where $c \geq -1/2$ is classified as *good* because the strongest known theorems fix $k/2$ bits of information to transmit the backdoor information (Theorem 6.3.1 and Corollary 6.1.8). There case where $-3/2 < c < -1/2$ is classified as *poor* because such theorems are typically used, but not in optimal ways. The case where $c \geq -3/2$ is classified as *failed* because at most $k/2$ bits of freedom in the key selection are kept. Often, an algorithm that generates only one backdoored key can be trivially modified via the precedingly cited theorems to free $k/2$ bits of information. Typically, this is the source of cardinality of such an algorithm.

Rating	value of c
Failed	$c \leq -3/2$
Poor	$-3/2 < c < -1/2$
Good	$c \geq -1/2$

Table 5–1: Cardinality classification for RSA backdoored key generators.

Cardinality classification for EG backdoored key generators. Just as for RSA, follow the relative ratings for EG backdoored key generators, in increasing order of desirability. (The EG key generator is given in Figure 7–11.) The security parameter of G_0 and G_1 is $k = |p|$. In practice, the ratio of the cardinality of G_1 to the one of G_0 is of the form $2^{c \cdot |p|} \cdot f(|p|)$, where $c \leq 0$ and $f(|p|)^{-1} \in 2^{o(|p|)}$.

Then, classes of values of c are meaningful. When from a backdoored generator, the value of a is dependent on p and α, so this fixes $|p|$ bits of information, because the value of an honest a (of length $|p|$) is independent from other parameters. Therefore, cases where $c \geq -1$ are classified as *good*. Depending on how the information that allows the retrieval of a is stored within the other parameters, some additional bits become fixed. Therefore, cases where $-2 < c < -1$ are classified as *average*. (This contrast with the *poor* label given for the intermediary case for RSA, because in all known *average* cases, there is no non-optimal use of theorems issue.) Finally, cases where $c \leq -2$ are classified as *failed*. This means that $2|p|$ bits are fixed in order to retrieve $|p|$ bits, and this happens only when an additional parameter is kept fixed. (While it may be useful in some situations, this usefulness is typically special to EG and thus is not included in our scheme of analysis.)

Rating	value of c
Failed	$c \leq -2$
Average	$-2 < c < -1$
Good	$c \geq -1$

Table 5–2: Cardinality classification for EG backdoored key generators.

5.1.3 Distribution properties

In general, the bits generated by G_0 are distributed in a special way which is distinguishable from uniformly distributed random bits. Therefore, the goal of the designer is to avoid any disturbance in this distribution. Three aspects are considered:

1. the number of bits occupied by embedding the encrypted backdoor information;
2. the positions of these bits;
3. the distribution of these bits w.r.t. their distribution when generated by G_0.

Number of bits occupied by the embedding. This is addressed as a part of the cardinality of KSM, in Subsection 5.1.2.

Distribution of the information via the embedding. Intuitively, the more spread out the random bits by the embedding function, f_R, the better, assuming that the embedding is within the bits of the public key which are distributed as the encrypted backdoor information is (assumedly, distributed computationally indistinguishably from uniform). For this criterion, the worst instances of f_R embed the encrypted backdoor information in the same block of bits, for any choice of parameters. The best instances of f_R embed the encrypted backdoor information in bit positions that are themselves close to uniformly distributed, over the random parameters in G_0 that are independent of the backdoor information.

However, this intuition is incomplete. In Definition 4.2.5, Point 2 requires that E's output be close to uniform, upon certain conditions. In fact, one is rather interested in the uniformity of the composition of embedding function and the encryption function, $f_R \circ E$, as a whole. There are two questions. Firstly, has f_R or E sufficient properties to guarantee the uniformity of their composition? Secondly, do f_R and E interact in insecure ways?

Consider the following example of a simple backdoored key generator. Suppose that E is simple, for instance, let E be a one-time pad. Suppose that f_R fixes the bit positions, then E can efficiently be inverted after two uses. That these f_R and E commute makes this backdoored key generator insecure. Therefore, a simple f_R calls for the use of a cryptographically strong E.

In the opposite case, if f_R is a pseudo-random-like function on $I(k_{priv})$. Then the sufficient assumptions on E are weaker. In certain circumstances, it is more efficient to use simpler E and more complex f_R. Chapter 7 develops this intuition into a method for designing backdoored key generators.

Definition 5.1.9 (Entropy of G_1) *Denote the* **entropy of Algorithm** G_1 *by* $\mathcal{H}_{G_1}$. *Consider the effect of $g = f_R \circ E$ on $x = I(k_{priv})$. The distribution of $g(x)$ is taken over the other random variables in E and f_R, denoted y. Thus y is the secret key (or index) of g.*

$$\mathcal{H}_{G_1} = \begin{cases} 1 & \textit{if } g_y(x)\textit{'s output is computationally indistinguishable from uniform} \\ 0 & \textit{if } g_y(x)\textit{'s output is computationally predictable, given a polynomial sample.} \end{cases}$$

The entropy is denoted as

$$0 < \mathcal{H}_{G_1} < 1$$

if the embedding is not predictable but also is distinguishable from uniform.

Distribution w.r.t. the one when generated by G_0. The positions where the bits of backdoor information are inserted is a relevant factor in avoiding such disturbance. For instance, with $G_0 = \textsc{rsa}$, the middle bits of n are more randomly distributed. It is preferable to embed a backdoor away from the very least and most significant bits.

In G_1, assume that the distribution of the embedded encryption of the backdoor information is close to uniform. According to Definition 5.1.9, this is equivalent to assuming $\mathcal{H}_{G_1} = 1$. In G_0, if the b^{th} bit of $k_{p(G_0)}$ is not uniformly distributed, then it is a bad place for f_K to embed one of the bits of the encrypted backdoor information, $E \circ I(k_{priv})$. Including the b^{th} bit of $k_{p(G_0)}$ in k_B is not useful in order to satisfy the indistinguishability property (3.a) of Definition 4.2.9.

Therefore, one requires a measure on the distribution produced by G_0, in function of the positions determined by G_1, via its embedding function, f_R. Let the concatenation of these bits be denoted by X and their distribution, by $\mathcal{X}$.

Suppose that G_1's output is close to the uniform distribution, so if G_0's output is close to uniform, then the outputs of G_0 and G_1 are similar. One wishes to determine how close $\mathcal{X}$ is to the uniform distribution. This is given by the statistical distance between $\mathcal{X}$ and the uniform distribution $\mathcal{U}$, from which a $|X|$-bit length random variable, U, is sampled [Lub96, p.70].

Definition 5.1.10 (Uniformity correctness) *The uniformity correctness of G_1 is the statistical distance between $\mathcal{X}$ and $\mathcal{U}$.*

$$\mathcal{C}_{G_1} = \mathit{dist}(\mathcal{X},\mathcal{U}) = 1/2 \cdot \sum_{z \in \{0,1\}^{|X|}} \left| \Pr_X[X = z] - \Pr_U[U = z] \right|$$

Equivalently,

$$\mathcal{C}_{G_1} = \mathit{dist}(\mathcal{X},\mathcal{U}) = \max_{S \subseteq \{0,1\}^{|X|}} \left\{ \Pr_X[X \in S] - \Pr_U[U \in S] \right\}.$$

If $\mathcal{C}_{G_1}$ is negligibly small w.r.t. the security parameter of G_0 (which is, in general, equivalent to stating w.r.t. $|X|$), then the consequent parts of G_1 are close to being uniformly distributed, and therefore are statistically indistinguishable from G_0's. In this case, G_1 is said to satisfy the uniformity correctness criterion.

Distribution of G_0. Finding $\mathcal{C}_{G_1}$ can be a considerable problem. It requires the study of the distribution of the honest algorithms' public parameters, as in [CS03, Section 5] for the parameter n of RSA. A further study of the distribution of the product of two 512-bit primes (with a leading bit of 1) has been done by the author of this work. However, because of the approximation due to results related to the Prime number Theorem (Theorem 2.2.4), no conclusive new results were produced.

The two measures introduced in this section are sufficient to produce additional qualitative indications on backdoored generators. However, they are not sufficient in order to provide precise quantitative indications. Further study of the distributions produced by honest key generators would be required.

5.1.4 Generalized key regeneration

The user of an honest key generator expects the algorithm to regenerate keys at will, so that the distribution of the keys, as elaborated in the previous section, can be analyzed. Since the user may not fully trust the generator, the user may require the more general regeneration that consists in requesting keys whilst fixing some part of these keys. For instance, the RSA parameter e may be kept fixed and the n regenerated; one of the primes may be kept fixed and the rest regenerated, etc. Not to allow this power to the user would be similar to restricting the user to request the regeneration of only a "reasonably" small number of keys, as a non-distinguishing user should only need. Not complying to either type of regeneration is evidence that a backdoored key generator is used.

Obviously, this criterion only makes sense if the honest algorithm, G_0, can itself be adapted to satisfy it. Some G_0 may be defined as to generate key parameters that are correlated.

Definition 5.1.11 (Generalized key regeneration) *The requirements of* **generalized key regeneration** *(GKR) are as follows.*

1. *The distinguisher may ask for a new public key.*
2. *The distinguisher may ask to keep certain parameters of the public key (given that it has many parts) and request the remaining parameters afresh.*

Point 1 is reasonable, because the distinguisher should have a polynomial-size sample of keys in order to make a decision. Point 2 is reasonable, because elegant solutions exist where the backdoored key generator does not scramble parameters in unnatural or contrived ways. Although it is not stated as a standard requirement, any honest key generator complies with GKR.

Therefore, the backdoored key generator should be prepared for the distinguisher to be content with part of the public key, while discontent with the rest. In particular, any suspicious parameter should be regenerable at the will of the distinguisher. Complying with GKR implies that the backdoored key generator cannot ignore the independence relations between parameters, just as it cannot ignore issues regarding key distribution. This confirms the intuition that "randomly" distributed variables means variables that are uniformly and independently distributed, as far as the honest key generation requires it.

Summing up, GKR is a feature of a backdoored key generator. Note that this feature often seems to require the use of volatile memory (Section 5.2). This feature was first taken into account in [CS03].

Correlation between key components. As the distinguisher fixes a key component, the backdoored key generator may generate keys with components more correlated than they should be, according to the distribution produced by G_0.

For instance, if the encryption function, E, is deterministic, we have the following two cases.

Example 5.1.12 (Distinguishable variable correlation) Suppose the following situation: $G_0 = \textsc{rsa}$, $I(k_{priv}) = p$ and $k_B = e$. This means that p is concealed in e.

In this case, if p is fixed by the distinguisher, the parameter e may either be the same upon regeneration, or have a fixed part. GKR allows the distinguisher to discover a correlation between p and e which is not present in the honest key generation. $\square$

Example 5.1.13 (Indistinguishable variable correlation) Suppose $G_0 = \textsc{rsa}$ and $I(k_{priv}) = \delta$, small enough for Coppersmith's attack. The small δ is concealed as $e = \pi_\beta(\delta^{-1} \bmod \phi(n))$.

In this case, if p, q or d is fixed by the distinguisher, the generator outputs a different value of e (except negligibly often). GKR does not give a significant advantage to the distinguisher in this case. $\square$

From the two preceding examples, the following definition of correlation between parameters in backdoored public keys can be stated.

Definition 5.1.14 (Variable correlation) *Let the* **variable correlation** *of algorithm* G_1 *be denoted as* $\mathcal{V}_{G_1}$. *It is defined accordingly to whether the execution of* G_1 *is distinguishable from the one of* G_0 *via the agreement or refusal to comply with GKR.*

$$\mathcal{V}_{G_1} = \begin{cases} 0 & \textit{if it is distinguishable} \\ 1 & \textit{if it is indistinguishable} \end{cases}$$

5.1.5 Diversity

To assign a quality to the static analysis (without interactions such as side channel analyses) of a backdoored key generator G_1, define the key diversity as directly proportional to the key ratio (Definition 5.1.7). It is also directly proportional to one minus the uniformity correctness (Definition 5.1.10) as well as to the entropy (Definition 5.1.9). Finally, if the variable correlation (Definition 5.1.14) of Algorithm G_1 is null, then the diversity is null as well.

Definition 5.1.15 (Key diversity) *Denote the* **key diversity of Algorithm** G_1 *by* $\mathcal{D}_{G_1}$. *Then*

$$\begin{aligned} \mathcal{D} &= 0 \quad \textit{if} \quad \mathcal{V} = 0 \\ \mathcal{D} &\propto \mathcal{R} \\ \mathcal{D} &\propto \mathcal{H} \\ \mathcal{D} &\propto 1 - \mathcal{C}. \end{aligned}$$

Key diversity is a stronger measure than computational indistinguishability and it is more applicable. However, the above definition implies a set of possible measures of diversity. Nevertheless, the definition as stated is sufficient to compare the backdoored key generators that are analyzed in the following two chapters.

Conclusion. To prevent attacks based on distinguishable properties, it is also required that the generable keys' number and distribution be close to the ones of generable honest keys. The *key diversity* of a backdoored key generator is related to the number of backdoored keys that it can generate as well as to their distribution. It is not sufficient to generate a high number of backdoored keys, because this can be done in trivial ways which possess easily distinguishable properties (Subsection 7.2.2).

$$\cdot \begin{array}{c}\text{key diversity}\\ \text{(Definition 5.1.15)}\end{array} \begin{cases} \cdot \begin{array}{c}\text{number of keys}\\ \text{(Definition 5.1.7)}\end{array} \\ \\ \cdot\, \text{distribution} \qquad \begin{cases} \cdot \begin{array}{c}\text{uniformity correctness}\\ \text{(Definition 5.1.10)}\end{array} \\ \cdot \begin{array}{c}\text{entropy}\\ \text{(Definition 5.1.9)}\end{array} \\ \cdot \begin{array}{c}\text{variable correlation}\\ \text{(Definition 5.1.14)}\end{array} \end{cases} \end{cases}$$

An increase in the number of keys or in the key entropy implies an increase in diversity. This is coherent with Definition 5.1.15. Conversely, one would like to state that a greater diversity cannot decrease any type of indistinguishability. However, diversity being a syncretism of different versions of indistinguishability, no such statement can be made. Alike to the preceding affirmation concerning the number of keys and diversity, only hypotheses on a type of indistinguishability can induce conclusions on diversity.

Key diversity is one way to capture the static nature of the keys, up to symmetry or asymmetry. This last characteristic is stated separately.

5.2 Interactions with the generator: side channel analyses

Some interactions normally exist between a distinguisher and an honest key generator. These interactions can be measured and used to distinguish between a malicious and an honest generator. This is a dynamic type of indistinguishability, as opposed to the static indistinguishability presented in Section 5.1. Therefore, the formal goal of the dynamic measures is to evaluate how an algorithm satisfies Property 3(b) of Definition 4.2.9.

The distinguisher may analyze the resource consumption of a key generator. If an algorithm appears to take a longer time or to use more memory, or a special type thereof, then it is likely that it is dishonest. A very simple distinguisher can determine this. Ideally, a backdoored key generator should use the same resources as the honest one.

5.2.1 Complexity

Relevance of this measure. The complexity of the backdoored key generator is a feature that can be analyzed through side channel analysis. By opposition, it is not relevant to analyze the complexity of the retrieval of the private keys (leaked via the backdoored public keys) beyond insuring polynomial time. side channel analysis only applies to the key generator, because this is the only one of the designer's activities that involves the legitimate user (who plays the role, which was invented for analysis purposes, of the external distinguisher).

Ideally for the designer, backdoored key generation would display the same complexity as the honest key generation. However, all backdoored key generators published so far involve additional operations. Intuitively, this appears to be an intrinsic feature of backdoored key generators, because backdoored keys have additional properties as compared to honest keys, one would expect backdoored keys to be generated by an algorithm that performs additional operations. However, a backdoored key generator that displays exactly the same complexity can be trivially constructed, at the cost of generating less keys, pooled from a table, for instance.

Classification scheme. In order to classify backdoored key generators with respect to the side channel analysis of time complexity, we compare the complexity of backdoored and honest key generators. One way to compare the complexity of G_1 and G_0 is to express them as functions of the security parameter of the cryptosystem which the generation belongs to.

Another way is to ignore the operations which do not take a significant time complexity, i.e. to pick out the most expensive operations that G_0 performs. Then, the analysis counts the number of times G_1 uses these operations. In this way, only significant operations are considered, i.e. the ones that the distinguisher would find significant.

An additional idea applies to the whole of complexity analysis. The complexities in generating separate parameters of a public key are considered separately, because of their separate examination via GKR, as in Subsection 5.1.4.

Application to RSA. In this context, the interesting notion of complexity is relative to the one of the honest key generation algorithm for RSA. The most expensive operations of G_0 are the generation of two random primes of size k. Then checking $\gcd(x, \phi(n)) = 1$, for a given x, along with inversions modulo $\phi(n)$, are the next most costly operations.

```
Algorithm $G_0$ [Standard RSA key generation]
1: Pick random primes $p, q$ of appropriate size $= k$, and set $n = pq$.
2: repeat
3:    Pick a random odd $e$ such that $|e| \leq 2k$.
4: until $\gcd(e, \phi(n)) = 1$.
5: Compute $d \equiv e^{-1} \bmod \phi(n)$.
6: return $K = (p, q, d, e)$.
```

Figure 5–2: Honest RSA key generation (recall of Figure 2–5).

Denote as t_n the expected time complexity of generating the parameter $n = pq$. It holds that $t_n = 2t_p = 2t_q$, where t_p and t_q are the expected time complexity of generating random primes of size k, ignoring the time to perform multiplication. More precisely, t_p is equal to the time of generating the bits of p multiplied by the expected number of times this operation is repeated in order to find a prime. If constants are considered irrelevant, then the complexity for p or q needs only to be expressed in terms of t_n: with this abuse of notation, $t_p = t_q \approx t_n$.

With another abuse of notation, also denote as t_e the expected time complexity of generating the pameter e, but given that n has already been generated. Consider two quantities, the probability P of selecting a random element of $\mathbb{Z}^*_{\phi(n)}$ and the expected time E, of an inversion modulo $\phi(n)$. In G_0, these appear to be performed consecutively, as Steps 2 to 4 and Step 5 respectively. However, using the extended Euclidian algorithm, the two operations are done at the same time. The probability that an inverse exists is P and given it does, the time to find it is E. Therefore,

$$t_e = (1/P) \cdot E = E/P.$$

Overall, the complexity of G_0 can be denoted as:

$$T(G_0) = t_n + t_e. \tag{5.2}$$

Comparing complexities. Up to now, the best RSA backdoored key generators have an expected complexity that can be said to be *linear in the one of the honest generator*. More precisely, the best achievable complexity of an algorithm, G_1, is proportional to a constant times each of the complexities of the prime generation and the one of checking gcd's or inverting modulo $\phi(n)$, separately:

$$\begin{aligned} t_n(G_1) &= c_1 t_n, \\ t_e(G_1) &= c_2 t_e, \end{aligned}$$

where c_1, c_2 are given constants. For $G_0 =$ RSA, it is important to consider these separately. If e is regenerated (GKR), then t_e is observed separately, while in Equation 5.2, the dominant term is t_n. Therefore, if complexity was analyzed only in the form of Equation 5.2, then the total complexity would not be easily affected by detectable, but not dominating, variations in t_e.

A linear complexity allows a backdoored key generator, G_1, to maintain its security as the security parameter is increased. If the security regarding the side channel analysis of complexity can be assured for a given value of the security parameter, then it can be assured for all other values, with a constant amount of additional computational power. For instance, suppose that $T(G_1)$ is exactly $2T(G_0)$, without ignoring constants, as we otherwise do. If the generator is a hardware device, its circuitry's computational power has to be doubled to match exactly a complexity of $T(G_0)$. Ignoring constants again, the first backdoored key generator to have a complexity of $T(G_0)$ was shown in our work (Chapter 7).

Likewise, an RSA backdoored key generator may have an expected complexity that is said to be *polynomial in the one of the honest generator*. This means that $T(G_1)$ is proportional to a polynomial in t_n or t_e. There are many examples of quadratic complexity. For the above reason, these complexities compromise a backdoor's security. Nevertheless, some such backdoors are theoretically interesting in providing intuition to design more efficient ones.

Complexity classification for RSA backdoored key generators. Follow the relative ratings for RSA backdoored key generators, in increasing order of desirability. Denote by t the complexity of G_0 for a given key parameter. In practice, the complexity of G_1 is of the form

$$T(G_1) = t^a \cdot T(F)^b + t^c$$

where a, b and c are constant exponents and F is a non-instanciated function such as π_β at the end of this section. Because $T(F)$ may or may not be a dominant factor, complexities with $b > 0$ are then still "to be evaluated", so they are rated as F-*relative*, by convention. For instance, if $F =$ AES is called within the generation of a prime p, then for any small constant b, the factor $T(F)^b$ is negligible with respect to t_p.

Rating	complexity
Failed	The complexity is at least *quadratic* in the complexity of G_0. In other words, $a \geq 2$ or $c \geq 2$.
	The function F is explicit and such that the factor $T(F)^b$ is non-negligible with respect to t.
Poor	The complexity is more than *linear* but less than quadratic in the complexity of G_0. In other words, $1 < a < 2$ and $1 < c < 2$.
F-relative	The complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly (depending on F, the factor $T(F)^b$ may or may not be dominant). In other words, $a < 2$, $b > 0$ and $c < 2$.
Good	The complexity is *linear* in the complexity of G_0, and F is explicit and such that the factor $T(F)^b$ is negligible with respect to t. In other words, $a \leq 1$, $b \geq 0$ (small) and $c \leq 1$.

Table 5–3: Complexity classification for RSA backdoored key generators.

Application to EG. The complexity is accounted w.r.t. the one of EG key generation, G_0, as given in Figure 5–3. Its most costly operations are the generation of a random prime of size $|p|$ and the loop which checks whether α is a generator modulo p. Then the complexity of G_0 can be denoted as:

$$T(G_0) = t_p + t_\alpha.$$

and, as for $G_0 =$ RSA, t_p and t_α are analyzed separately. In this case as well, t_p is the dominant term.

Complexity classification for EG backdoored key generators. Just as for RSA, the complexity of G_1 is of the form

$$T(G_1) = t^a \cdot T(F)^b + t^c$$

```
Algorithm $G_0$ [Standard EG key generation]
1: Pick a random prime $p$ of appropriate size.
2: repeat
3:     Pick a random $\alpha \in \mathbb{Z}_p^*$.
4: until $\forall$ prime $p_i$, s.t. $p_i | p-1$, it holds that $\left[\alpha^{(p-1)/p_i} \bmod p \neq 1\right]$.
5: Pick a random $0 \leq a \leq p-2$.
6: Compute $\beta = \alpha^a \bmod p$.
7: return $(p, \alpha, a, \beta)$.
```

Figure 5–3: Honest EG key generation (same as Figure A.1).

where a, b and c are constant exponents and F is a non-instanciated function such as π_β at the end of this section.

Follow the relative ratings for EG backdoored key generators, in increasing order of desirability.

Rating	complexity
Failed	The complexity is *quadratic* in the complexity of G_0. In other words, $a \geq 2$ or $c \geq 2$.
	The function F is explicit and such that the factor $T(F)^b$ is non-negligible with respect to t.
Poor	The complexity is less than quadratic in the complexity of G_0 times the one of a cryptographic function F, such that $T(F)$ is negligible. In other words, $a < 2$, $b = 1$ and $c < 2$.
F-relative	The complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly (depending on F, the factor $T(F)^b$ may or may not be dominant). In other words, $a < 2$, $b > 0$ and $c < 2$.
Good	The complexity is *linear* in the complexity of G_0 plus the one of a cryptographic function F, such that $T(F)$ is negligible, or better. In other words, $a = 0$, $b \leq 1$ and $c \leq 1$.

Table 5–4: Complexity classification for EG backdoored key generators.

Additional notation. Another common case is when $T(G_1)$ is proportional to t_n or t_e times the complexity of an unknown, non-instantiated, function F. This function typically is a cryptographic primitive. The notation applies for a pseudo-random function, $F = \pi_\beta$, keyed with β. As before, the expected time complexity of π_β is denoted as

$$T(F) = T(\pi_\beta).$$

5.2.2 Memory

Relevance and classification scheme. Memory usage of the backdoored key generator is another feature that can be analyzed through side channel analysis. One can distinguish two types of memory that may be used, volatile memory (VM) and non-volatile memory (NM). Some backdoored key generators require NM in order to transmit the necessary information from one run to another. Others require VM in order for key regeneration to function.

The use of VM is preferable to the use of NM, because the former is more discreet. It could appear that NM requires more resources, so it is more easily detected. However, in practice NM is resettable anyway. If a backdoored key generator is dependent on NM to remember the keys that have been generated in the past, such a reset can make it regenerate the same past keys. This is easily detectable.

Results. In some of the algorithms of [YY97b, YY05a], non-volatile memory is used. Indeed, in order for the backdoor to leak all the appropriate information, two emissions of backdoored keys are required. Thus, the first one is remembered (even if the device is turned off, requiring the non-volatile quality of the memory), in order for the second one to recover the backdoor or to generate it securely. More details are given in Section 6.2, and more particularly in Subsections 6.2.2 and 6.2.3.

Example 5.2.1 (NM for backdoor recovery) In Figure 6–9 (Subsection 6.2.2), a first message allows the transmission of an exponent, c. A second message transmits a function of this exponent, $f(c)$, such that given the two messages and a private key, the designer can recover a given function of the exponent, $g(c)$.

The designer needs to preserve the first message in non-volatile memory (NM). The second message and the private key alone are not sufficient to recover $g(c)$. $\square$

Example 5.2.2 (NM for backdoor security) In Figure 6–11 (Subsection 6.2.3), a counter (the variable i) is stored in non-volatile memory. This is a counter for the number of backdoored keys that the generator created.

The non-fixed seed to the pseudo-random generator is $u:i:j$. The variable u is randomly chosen in $\mathbb{Z}_N^*$, where N is the designer's Rabin public key (u is a seed to generate the backdoor information). The counter j is reset for every generated key (it is incremented until some condition is satisfied). The counter i insures that the generator's pseudo-random seed is necessarily different. Thus the same keys are at least as unlikely to be regenerated as it is unlikely to find a collision in the keyed pseudo-random generator, f_K:

$$\mathrm{key}_i = f_K(u:i:j)$$

and

$$\mathrm{key}_{i+1} = f_K(u:i+1:j).$$

Therefore, NM is required for security because of key regeneration, despite its weaknesses as pointed out earlier in this section. $\square$

Backdoor algorithms that allow generalized key regeneration (GKR, as defined in Subsection 5.1.4) sometimes appear to require non-volatile memory. This seems to be the case for backdoored EG key generation, because these keys' parameters are more interdependent than RSA keys' parameters. An example is our algorithm in Figures 7–12 and 7–13 (Subsection 7.3.2).

Example 5.2.3 (VM for backdoor security) The algorithm in Figure 7–13 answers the regeneration (GKR) of the private key parameter a of the EG cryptosystem, while keeping all other parameters constant. The variable r is a random seed used to pseudo-randomly generate some of the parameters, including a, so r is stored in volatile memory as to allow backtracking: the previous generations' essential internal variables are preserved in r. These variables r and s are the seeds for the generation of another EG parameter, α. Informally,

$$a = f_K(r)$$

and

$$\alpha = f_K(s:r)$$

for an invertible keyed pseudo-random function f_K.

For this algorithm, VM is required for security because of GKR. For instance, in the case where the legitimate user wants to keep a and regenerate α, the new α may be regenerated with the same r and a different value for s. Both pairs of parameters, old and new, are coherent with having been generated from the seed r.

This coherence, along with the invertibility of f, insures completeness:

$$s\!:\!r = f_K^{-1}(\alpha)$$

and

$$a = f_K(r).$$

So the designer retrieves the private a. $\square$

For RSA backdoored key generation, generalized key regeneration does not require any memory. For symmetric backdoors, this is shown by the algorithms of [CS03], in Section 6.3. For asymmetric backdoors, this is shown by our algorithms, in Section 7.1.

5.3 Measures on the design of algorithms

We briefly elaborate on two additional measures that relate to the design of algorithms.

5.3.1 Computational assumptions

The underlying cryptosystem associated to G_0 introduces some computational assumptions. For instance, RSA assumes that the e^{th} root modulo n is difficult to compute.

The designer of a backdoored key generator wishes to conceal the information that is useful to retrieve the private key, which is embedded in the public key. To achieve this, i.e. the confidentiality and indistinguishability of Definition 4.2.9, the designer relies on some assumptions about computational infeasibility. On the contrary, completeness is independent of such computational assumptions, as it concerns the capacity of key retrieval, i.e. the feasibility of a computation.

For instance, in the case of RSA, the backdoored key generator may reuse the assumption that the e^{th} root modulo n is not feasibly computable. In the case of ElGamal over elliptic curves, the backdoored key generator may reuse the assumption that the discrete logarithm problem is hard on average over elliptic curves.

As detailed in Subsection 7.2.3, for symmetric backdoored key generators, it can even be the case that the permutation π_β is so simple that it requires no computational assumption. For instance, π_β can be as simple as xoring or shuffling its input bits.

In our comparative study of existing and new backdoored key generators, additionally used computational assumptions are explicitly mentioned. The extra computational assumptions shall be as minimal as possible.

5.3.2 Simplicity

Another consideration is the simplicity and parsimony of a dishonest generator. Typically, simpler algorithms are more flexible, more easily implementable, require less additional computational assumptions, their properties can be more easily analyzed, and overall less possibilities for errors are introduced. *Lex parsimoniae*: entities should not be multiplied beyond necessity, the principle for scientific theories known as Occam's razor.

Algorithms can be rated as having low structure (LS), medium structure (MS) or high structure (HS), structure being the opposite of simplicity.

5.4 Measures: summary

Table 5–5 gives the properties and a list of measures, with dependencies, relevant to a backdoor.

To every pair of backdoored key generator and retrieval algorithm, (G_1, R_1), is associated such a table. Such a table is a wieldy way of summarizing their properties, as opposed to enumerating them.

confidentiality			
completeness			
indistinguishability	asymmetry		
	diversity	ratio $\mathcal{R}_{G_1}$	
		distribution	uniformity ($\mathcal{C}_{G_1}$)
			entropy ($\mathcal{H}_{G_1}$)
			variable correlation ($\mathcal{V}_{G_1}$)
	generalized key regeneration (GKR)		
	side channel analyses		complexity
			memory (VM or NM)
computational assumptions			
simplicity (LS, MS, or HS)			

Table 5–5: Properties and measures relevant to a backdoor, with dependencies.

An additional column is presented with each algorithm's properties, which assigns a qualitative value to each property. Grades of "Good" "Average", "Poor", and "Failed" rate the measures. As mentioned before (in Subsection 5.2.1), the rating *F-relative* is also used when an algorithm leaves undefined determining features, such as cryptographic functions.

5.5 Chapter notes

Chapter 5. Measures for backdoored keys. The interactions with the generator that we study do not include zero-knowledge protocols because ZK is not standardly included in known and used key generators. The possible interactions with a dishonest generator that we consider are the same than that with an honest one. However, note that zero-knowledge is used for RSA key generation with verifiable randomness [JG02].

Subsection 5.1.1. Classical definitions of indistinguishability. Kleptography uses polynomial indistinguishability in the sense of [GM84]. Shamir's backdoor in the public-key knapsack cryptosystem known as *Basic Merkle-Hellman* is an example of perfect indistinguishability: the backdoor breaks the cryptosystem [Sha82]. We will introduce *key diversity*, which relates to computational and statistical indistinguishabilities.

Related publication drafts. Key diversity was first introduced in [ACS]. A draft of this chapter was submitted for publication as [Arb].

CHAPTER 6
Comparison of existing algorithms

This chapter surveys existing backdoored key generators of theoretical interest, i.e. which are provided with an analysis and rest on standard computational assumptions. The goal of this chapter is to uniformly analyze them w.r.t. the criteria introduced in the preceding chapter and summarized in Table 5–5. This creates a basis of comparison for the new algorithms of Chapter 7.

6.1 Simple algorithms

The Anderson-Kaliski [1] and Howgrave-Graham [2] backdoors are simple examples of backdoored key generators. They provide useful intuition as to how to, as well as how to not, design such algorithms.

Both generators leave confidentiality unsatisfied, for different reasons. Confidentiality constitutes the main problem of the simple approach of the Anderson-Kaliski backdoors. For the Howgrave-Graham backdoors, the relatively simple approach also introduces more subtle issues w.r.t. key diversity and computational assumptions.

[1] Named after two people, Ross Anderson and Burton S. Kaliski Jr.

[2] Named after Nick Howgrave-Graham.

6.1.1 Anderson-Kaliski backdoors

An early theoretically interesting backdoored key generator was proposed by Anderson in [And93]. This symmetric RSA backdoor with fixed secret key A embeds in n an integer, qq', which is easy to factor. This integer is retrieved as $n \bmod A = qq'$. In turn, the knowledge of these factors, q and q', and of A allows to factor n itself.

The prime generation in Steps 1 to 4, as well as Steps 5 to 8, is claimed to be equivalent to the usual operation of selecting a random prime, $p \in_U \mathbf{rp}$. This is detailed in the paragraph *Distribution properties* which follows.

PARAMETERS:
- A is a fixed 200-bit prime number.
- Public $r_A, r'_A : \{0,1\}^{100} \rightarrow \{0,1\}^{56}$ are (assumed) pseudo-random functions.

```
Algorithm And93 [RSA key gen]
 1: repeat
 2:     Pick rp $q$ of size 100.
 3:     Set $p = r_A(q) \cdot A + q$ likely prime [Th. 6.1.5].
 4: until $p$ is prime.
 5: repeat
 6:     Pick rp $q'$ of size 100.
 7:     Set $p' = r'_A(q') \cdot A + q'$ likely prime [idem].
 8: until $p'$ is prime.
 9: Set $n = pp'$.
10: repeat
11:     Pick a random odd $e$ such that $|e| \leq 2k$.
12: until $\gcd(e, \phi(n)) = 1$.
13: Compute $d \equiv e^{-1} \bmod \phi(n)$.
14: return $(p, p', d, e)$.
```

```
Algorithm A-And93 [Key retrieval]
1: Input of $(n, e)$.
2: Set $n' = n \bmod A$.
3: Compute $q, q' = factor(n')$ [feasible: $|n'| = 200$].
4: Compute $p = r_A(q) \cdot A + q$.
5: Compute $p' = r'_A(q') \cdot A + q'$.
6: Compute $d \equiv e^{-1} \bmod \phi(n)$.
7: return $d$.
```

Figure 6–1: Anderson's backdoor for RSA. The factorization of n is hidden in n itself.

Confidentiality. The confidentiality of Anderson's backdoor was quickly broken by Kaliski in [Kal93]. For simplicity, denote $r_A(q)$ by r and similarly for r'. The distinguisher can break Anderson's backdoor by computing the continued fraction expansion of p'/p in which is found r'/r, by the following well-known theorem.

Theorem 6.1.1 (Continued fraction expansion) *Consider the continued fraction expansion of p'/p and a fraction r'/r. If*

$$\left|\frac{r'}{r} - \frac{p'}{p}\right| < \frac{1}{2r^2},$$

then r'/r is a convergent of the expansion of p'/p.

Thus $r_A(q)$ and $r'_A(q')$ are easily found. Theorem 6.1.1 is the same as Theorem 6.3.1, with renamed variables. The latter is followed by a sketch of proof.

Remark 6.1.2 *In Steps 3 and 4 of Algorithm And93, the length of the prime is*

$$|r| + |A| = 56 + 200 = 256$$

therefore n is 512-bit long.

Remark 6.1.3 *In the Anderson backdoor, the conditions of Theorem 6.1.1 are likely to be satisfied, by simple algebra. They are satisfied if*

$$2r(r'q - rq') < rA + q$$

which is most of the time, because the LHS is 213-bit long and the RHS, 256-bit long. If r is random, the conditions are satisfied with probability $1 - 1/2^{43}$.

Furthermore, if A is the same for all distinguishers (as it is the case in the original algorithm), then it can be retrieved by any party through solving simultaneous Diophantine equations via the LLL-reduction algorithm [LLL82]. The problem is reduced to finding a short vector in a lattice, which has not been proven to be NP-hard. Also, there are polynomial time algorithms that find relatively short vectors.

Completeness. Algorithm And93 embeds in n an integer, $n' = qq'$, which is easy to factor. This factorization, (q, q'), is the key to the factorization of $n = pp'$. The private key, A, is used to retrieve the integer, $n' = qq'$, as $n \bmod A = qq' = n'$.

Remark 6.1.4 *Step 2 of Algorithm A-And93 yields q and q' because*

$$n \bmod A = (r_A(q) \cdot A + q)(r'_A(q') \cdot A + q') \bmod A = qq'.$$

Then $(q, q') = factor(n')$ is feasible because $|n'| = 200$. The knowledge of these factors, q and q', and of A allows to factor $n = pp'$. Steps 4 and 5 of Algorithm A-And93 perform this computation, using the public functions r and r'.

$$\begin{aligned} p &= r_A(q) \cdot A + q \\ p' &= r'_A(q') \cdot A + q'. \end{aligned}$$

Symmetry. The Anderson-Kaliski algorithm is symmetric, with secret key A.

Kaliski suggested how the Anderson backdoor could be improved. By letting the A parameter be distributed randomly as an additional Step 0 in Algorithm And93, only the distinguisher could deduce its own A, because only the distinguisher could hold simultaneous Diophantine equations with the same A. However, in this case, indistinguishability fails, and the backdoor is still considered broken.

Note that the original Anderson backdoor, where A is fixed, is consistent with our definition of symmetric backdoored key generator, while Kaliski's version, where A is distributed randomly, is not.

Cardinality. In order to compare this algorithm with the ones that follow, the translation of the bit sizes to asymptotic notation is necessary. From Remark 6.1.2

and Steps 1 to 4 of Algorithm And93, $|q| = |q'| \approx |p|/2 = |p'|/2$ so $|n'| \approx k$. From Steps 1 and 2 of the algorithm,

$$\#\{p\} = \#\{p'\} = \#\{q\} = \#\{q'\} \approx 2^{k/2-\lg(k/2)}$$

therefore

$$\#\{n\} = \#\{n'\} = \#\{q\} \cdot \#\{q'\} \approx 2^{k-2\lg(k/2)}.$$

Recall, from Example 2.2.9, that there are about $2^{4k-2\lg k}$honest RSA pairs (n, e). There are about $2^{2k} = n$ public exponents e, in both honest and dishonest cases, so that the factor in e cancels in the ratio of honest to dishonest keys.

$$\mathcal{R}_{G_1} = \frac{2^{k-2\lg(k/2)}}{2^{2k-2\lg k}} = 2^{-k+\lg 2} = 2^{-k} \cdot 2 \approx 2^{-k}$$

The ratio of honest to dishonest keys is approximately 2^{-k}. This is because the two logarithms (in exponent) cancel to a factor of 2.

Distribution properties. It is claimed in the original papers that the parameter $n = pp'$ is likely to be a product of two primes, because of Theorem 6.1.5.

Theorem 6.1.5 (Dirichlet's Theorem in Number Theory) *For all co-prime positive integers q and A, there are infinitely many prime numbers congruent to q mod A.*

This theorem implies that there are as many primes of this form as there are prime numbers (both sets are countable). However, it is not strong enough, because it ignores distribution. The random numbers that are picked are multiples of A, translated by q. Since this selection is independent, via the dominating term, $r \cdot A$, of the distribution of primes, it seems reasonable to claim that this selection is very close or equivalent to a uniformly random selection in a subset of positive integers. Although it does not prove it, Theorem 6.1.5 does not infirm this claim and thus supports it, as the cardinality of primes in this selection is consistent.

Remark 6.1.6 *In Steps 2 and 3 of Algorithm And93, p is likely to be prime because of Theorem 6.1.5. Its conditions are satisfied : A and q are co-prime because they are both prime numbers and $q < A$.*

However, this does not guarantee that the primes are generated as to possess the form that is standard for RSA primes, that is, with a leading bit of value 1. The algorithm would have to be modified so that only half of the p and p' are accepted, that is, those with such leading bit. Overall, the distribution properties are good, given the pseudo-randomness of r and r', which are not explicitly provided.

Entropy of embedding. Without loss of generality, consider the distribution of the embedding in p (the same applies to p'). The bits of q are randomized through the pseudo-random r. So there are $|r| = 56$ bits of information in p that are close to uniform and dependent on q. Via the selection of q, the $|q|$ lower bits of p (about half) are uniformly distributed. Therefore, one expects about $|r|/2 + |q|$ bits of p to be close to uniformly distributed, because about half of r is redundant, as it affects what is already affected by q. Therefore, the distribution of the embedding is not entirely uniform (with the selection made in the set of all appropriate length primes). This is denoted as $0 < \mathcal{H}_{G_1} < 1$. This entropy rests on the assumption of the pseudo-randomness of r and r', which are not explicitly provided.

Generalized key regeneration and variable correlation. This property is satisfied as parameters are generated independently: p depends on q and p', on q'.

Complexity. The time complexity of the generation of p by G_1 is approximated as follows. For clarity, denote the complexity for the generation of an $k/2$-bit prime, p or q, by $t_{\sqrt{n}}$. Then $k/2$ other bits are generated through the application of the function r, accounting for an additional complexity of $T(r)$. Finally, the resulting k bits are tested for primality and the process is repeated until it is successful. The number of repetitions are the cost of the test, minus the random bits generation, is exactly t_p. Because $t_p \approx t_{\sqrt{n}}$ (Subsection 5.2.1, paragraph *Application to* RSA), the generation of p by G_1 is about $(t_{\sqrt{n}} + T(r)) \cdot t_p = (t_{\sqrt{n}} + T(r)) \cdot t_n$, and similarly for p'. The rest of G_1 is comparable to G_0.

Overall,

$$\begin{aligned} t_n(G_1) &\approx (t_{\sqrt{n}} + T(r)) \cdot t_n \\ &\in \Omega\left(t_n^{3/2}\right) \end{aligned}$$

and

$$t_e(G_1) = t_e.$$

This complexity is rated as “poor” because it is more than linear but less than quadratic in the complexity of G_0 (assuming that $T(r) \in \mathcal{O}(t_{\sqrt{n}})$).

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses two pseudo-random functions, r and r', which are not explicitly provided.

Simplicity. Intuitively, the algorithm has a relatively low structure, even without comparing with other algorithms. It consists of almost the same steps as G_0.

confidentiality			no: Kaliski [Kal93]	**failed**
completeness			yes: Remark 6.1.4	good
indisting.	asymmetry		no: symmetric key A	
	diversity	$\mathcal{R}_{G_1}$	2^{-k}	poor
		$\mathcal{C}_{G_1}$	0, given r and r' (see [3])	average
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, given r and r'	poor
	GKR		yes	good
	side-channels	complexity	$t_n(G_1) = \Omega\left(t_n^{3/2}\right)$ $t_e(G_1) = t_e$	poor / F-relative [4]
		memory	no	good
computational assumptions			pseudo-random r and r'	average
simplicity			LS	good

Table 6–1: Properties of Algorithm And93 of Figure 6–1. Our trivial modification detailed in *Distribution properties* is accounted for.

[3] More precisely, given the pseudo-randomness of r and r'.

[4] Assuming that $T(r) \in \mathcal{O}(t_{\sqrt{n}})$, the complexity is rated poor. Without this assumption, the complexity is rated as *F-relative.*

6.1.2 Howgrave-Graham backdoors

In [HG01, p. 52], Howgrave-Graham mentions the two first simple backdoored key generators for RSA that embed $p]^{k/2} = p]^{|p|/2}$ in the public e, as shown in Figures 6–2 and 6–3. Although not explicit in the original formulation, we let the designer use a secret key β and a keyed cryptographic permutation, π_β, that is assumed to be a one-way function to insure confidentiality and a pseudo-random function to insure distribution and entropy properties (this is a trivial modification). Completeness is insured via one of the Coppersmith factorization methods which uses approximate values of p: an approximate p allows the factoring of n if the additive error is on the lower-half bits of p.

First, recall Coppersmith's Theorem. The proof of Theorem 6.1.7 rests on the LLL lattice basis reduction.

Theorem 6.1.7 ([Cop96], Theorem 3) *Let $f(x, y)$ be a polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately, and assume the coefficients of f are relatively prime as a set. Let X, Y be bounds on the desired solutions x_o, y_o. Define $\tilde{f} := f(Xx, Yy)$ and let D be the absolute value of the largest coefficient of $\tilde{f}$. If*

$$XY \leq D^{2/3\delta}$$

then in time polynomial in $\lg D$ and 2^δ, we can find all integer pairs (x_o, y_o) with $f(x_o, y_o) = 0$, $|x_o| \leq X$, $|y_o| \leq Y$.

A well-known corollary to this theorem is of use. We present its proof for comparison, as similar corollaries are derived in this work, for other algorithms.

Corollary 6.1.8 ([Cop96], Corollary 2) *Given an approximation p_o of p such that $|p - p_o| \leq n^{1/4}$, it is possible to factor $n = pq$ in time polynomial in k, that is,* $\lg(n)$*. Recall that it is assumed that $q < p < 2q$.*

proof: Let

$$\begin{aligned} p_o &= p + x_o \\ q_o &= \lfloor n/p_o \rfloor \stackrel{\text{def}}{=} q + y_o. \end{aligned}$$

Then

$$\begin{aligned} X = |x_o| &\leq n^{1/4} \\ Y = |y_o| &\leq \left|\frac{n}{p_o} - q\right| = \left|\frac{n}{p + x_o} - \frac{n}{p}\right| = \left|\frac{nx_o}{p(p + x_o)}\right| = \left|x_o \frac{q}{p_o}\right| \leq |x_o| \leq n^{1/4}. \end{aligned}$$

Let

$$f(x, y) = (p_o - x)(q_o - y) - n$$

then the solution to the zeros of $f(x, y)$ is (x_o, y_o). In Theorem 6.1.7, $\tilde{f}(x, y) = f(Xx, Yy)$ has $\delta = 1$ and

$$D = n^{3/4}.$$

The constant term $p_o q_o - n = x_o q + y_o p + x_o y_o$ is also of this size, as well as the coefficients of the first degree terms of $\tilde{f}$.

Thus, $D^{2/(3\delta)} = n^{1/2}$. Because $XY \leq n^{1/2}$, the solution (x_o, y_o) can be found efficiently. ■

Figure 6–2 shows how $p]^{k/2}$ is embedded in the public e, in such a way that the completeness is insured via Theorem 6.1.8. The public exponent e is a secretly permuted version of

$$\epsilon = K : p + r$$

where K is a random prefix. Therefore, by knowing the public pair (n, e), hence ϵ, the designer can compute p and q via the last theorem.

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \to \{0,1\}^{2k}$ is an assumed cryptographic permutation.

Algorithm HG-1 [RSA **key gen**]

```
1: Pick rp $p, q$ of size $k$ s.t. $q < p < 2q$.
2: Set $n = pq$.
3: repeat
4:     Pick random $K < n^{1/2}$ and $r < n^{1/4}$.
5:     Set $\epsilon = K : p + r$.
6:     Set $e = \pi_\beta(\epsilon)$.
7: until $\gcd(e, \phi(n)) = 1$.
8: Compute $d \equiv e^{-1} \bmod \phi(n)$.
9: return $(p, q, d, e)$.
```

Algorithm A-HG-1 [Key retr.]

```
1: Input of $(n, e)$.
2: Set $\epsilon = \pi_\beta^{-1}(e)$.
3: Compute $p, q$ from $(n, \epsilon)$ [Theorem 6.1.8].
4: Compute $d \equiv e^{-1} \bmod \phi(n)$.
5: return $d$.
```

Figure 6–2: Howgrave-Graham's first backdoor for RSA, which embeds p in e.

In Howgrave-Graham's second backdoor for RSA, p is scrambled with a multiplicative factor and an additive term, as shown in Figure 6–3 (the least significant half of the bits is randomized via the addition on the random r). To show completeness, the following corollary of Coppersmith's main theorem is derived.

Corollary 6.1.9 (Corollary of Theorem 6.1.7) *Given $p_o = k_o p + x_o$ for $|k_o| < n^{1/2}$ and $|x_o| < n^{1/4}$, it is possible to factor $n = pq$ in time polynomial in k.*

proof: Let

$$\begin{aligned} p_o &= k_o p + x_o \\ q_o &= \left\lfloor \frac{n}{p_o} \right\rfloor =^{\text{def}} k_o\, q + y_o. \end{aligned}$$

Then $y_o = q_o \bmod k_o$ gives the following bound on Y.

$$\begin{aligned} X = |x_o| &\leq n^{1/4} \\ Y = |y_o| &\leq |k_o| < n^{1/2} \end{aligned}$$

Therefore, $XY < n^{3/4}$. Let

$$f(x, y) = (p_o - x)(q_o - y) - n$$

then the solution to the zeros of $f(x, y)$ is (x_o, y_o). In Theorem 6.1.7, $\tilde{f}(x, y) = f(Xx, Yy)$ has $\delta = 1$ and $D = n^{3/2}$, so $D^{2/(3\delta)} = n$. Because $XY < n$, the solution (x_o, y_o) can be found efficiently. ■

Parameters:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \to \{0,1\}^{2k}$ is an assumed cryptographic permutation.

Algorithm HG-2 [RSA key gen]
```
1: Pick rp $p, q$ of size $k$ s.t. $q < p < 2q$.
2: Set $n = pq$.
3: repeat
4:     Pick random $K < n^{1/2}$ and $r < n^{1/4}$.
5:     Set $\epsilon = Kp + r$.
6:     Set $e = \pi_\beta(\epsilon)$.
7: until gcd$(e, \phi(n)) = 1$.
8: Compute $d \equiv e^{-1} \bmod \phi(n)$.
9: return $(p, q, d, e)$.
```

Algorithm A-HG-2 [Key retr.]
```
1: Input of $(n, e)$.
2: Set $\epsilon = \pi_\beta^{-1}(e)$.
3: Compute $p, q$ from $(n, \epsilon)$ [Corollary 6.1.9].
4: Compute $d \equiv e^{-1} \bmod \phi(n)$.
5: return $d$.
```

Figure 6–3: Howgrave-Graham's second backdoor for RSA, which embeds p in e.

Algorithms' properties. The analysis is done concurrently for both Howgrave-Graham algorithms, because they are similar, as to avoid unnecessary repetition.

The only difference with the first Howgrave-Graham backdoored key generator is Step 5 in the key generation and Step 3 in the key retrieval. In Step 5 of the generation, parameter K is not appended as a prefix in order to fill the length of ϵ, but instead it multiplies the value of p. Step 3 of the retrieval is modified accordingly, using Corollary 6.1.9.

Confidentiality. In HG-1, the backdoor is

$$K : p + r;$$

in HG-2, it is

$$Kp + r.$$

To achieve confidentiality in both cases, a one-way function, π_β, is applied to the bits corresponding to p.

However, if the permutation is applied exclusively on these bits, the backdoored keys are distinguishable because they all contain the same sub-string $\pi_\beta(p)$ (see *Computational assumptions* that follows). So π_β is composed with the entire f_{HG-1} or f_{HG-2} originally stated by Howgrave-Graham. In the settings of Subsection 5.1.1, these $\pi_\beta \circ f_{HG-1}$ and $\pi_\beta \circ f_{HG-2}$ are exactly $f_R \circ E \circ I$.

Overall, this property rests on an assumption on a function, π_β, which is not explicitly provided. Nevertheless, when confidentiality is satisfied, it is always the case that a computational assumption is used. The fact that an additional assumption is used is made explicit in *Computational assumptions*, which analysis follows.

Completeness. This property is insured by Corollary 6.1.9.

Symmetry. Assuming that a pseudo-random permutation, π_β, is used for confidentiality, the algorithm is symmetric with private key β.

Cardinality. This algorithm's strong point is its cardinality, which is close to the best known one. Recall, from Example 2.2.9, that there are about $2^{4k-2\lg k}$honest RSA pairs (n, e). There are about $2^{2k-2\lg k}$ public modulo n, in both honest and dishonest cases, so that the factor in n cancels in the ratio of honest to dishonest keys. There are $\mathcal{N}_{G_0,e} \approx 2^{2k} = n$ public exponents e in the honest case.

Compute the number of public exponents e in the dishonest case, $\mathcal{N}_{G_1,e}$, for a fixed p. Because the samplings of K and r introduce in total $\frac{3}{2}k$ random bits, there are $\mathcal{N}_{G_1,e} \approx 2^{\frac{3}{2}k}$ backdoored keys out of the 2^{2k} honest ones that are generable. For HG-2, this holds assuming that $f_{HG-2}(K, r) = Kp + r$ is approximately injective. For HG-1, $f_{HG-1}(K, r) = K\!:\!p + r$ is injective.

Compute the ratio of honest to dishonest keys, $\mathcal{R}_{G_1}$. The backdoor information, p, is embedded in e, so for each choice of parameter n, only one corresponding backdoored e is generable. In other words, a generated value of n ties the value of p to be used in the generation of e. Therefore, in the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that

$$\mathcal{R}_{G_1} = \frac{\mathcal{N}_{G_1,e}}{\mathcal{N}_{G_0,e}} \approx 2^{-\frac{1}{2}k}$$

where $\mathcal{N}_{G_1,e}$ is computed for a fixed p.

Distribution properties. The uniformity correctness, $\mathcal{C}_{G_1}$, is "good", if the RSA encryption exponent is uniform (no assumption from Subsection 2.2.8 is needed) and π_β is pseudo-random. Therefore, they are not computationally distinguishable from one another unless e is of a special form or the assumption on π_β fails. The distribution properties of the embedded backdoor information are good, given the pseudo-randomness of π_β, which is not explicitly provided.

Entropy of embedding. In both algorithms, the bits of p are close to uniformly spread in e, through the application of π_β. Discarding π_β, the entropy, $\mathcal{H}_{HG-1}$, would seem smaller than $\mathcal{H}_{HG-2}$, because f_{HG-2} spreads the random K into more bits of ϵ. For HG-1, the embedding of p has a uniform distribution via the addition of r, but the embedding's location is limited to the least significant half of the bits of ϵ. However, via π_β, the resulting embedding's distributions are the same for both algorithms, which is the only feature that affects entropy.

Generalized key regeneration and variable correlation. The values of p and e are linked so that generalized key regeneration is not possible. Because e is computed from p, their values vary accordingly together. Unless specific collisions in the pseudo-random permutation π_β can be found, the values of e and p cannot be regenerated independently. For instance, if e is kept fixed, p cannot be regenerated unless p' such that $\pi_\beta(p) = \pi_\beta(p')$ can be found. Therefore, the security of π_β is mutually exclusive with the generalized key regeneration of the Howgrave-Graham algorithms.

Complexity. The key generation steps are the same than for G_0, except for the loop for the generation of e. At each iteration, this loop involves an additional call to a pseudo-random permutation, π_β. Therefore, the complexity of this algorithm is

$$t_n(G_1) = t_n$$

and

$$t_e(G_1) \approx T(\pi_\beta) \cdot t_e.$$

The complexity is rated "good", because it is linear in the complexities of G_0 and π_β. As for the previous algorithm, because π_β is not instantiated, the complexity is also rated "F-relative".

Memory. No use of additional memory is made.

Computational assumptions. If the permutation, π_β, is applied exclusively on the bits corresponding to p, the backdoored keys are distinguishable because they may all contain the same sub-string $\pi_\beta(p)$. This would mean the failure of the *GKR and variable correlation* criterion. More precisely, suppose that p is kept fixed, therefore $\pi_\beta(p)$ is a constant. By subtracting $K_1 : \pi_\beta(p) + r_1$ from $K_2 : \pi_\beta(p) + r_2$, one finds $\pi_\beta(p)\rceil^{k/2}$, which is derivable from any pair of backdoored keys. A similar argument holds for the other version of this backdoor, using a gcd instead of a subtraction.

It is therefore required to use some of the other bits of the backdoor as random seeds in π_β. This moves the indistinguishability problem from the properties of the functions, f_{HG-1} and f_{HG-2}, to that of the security of π_β. In our modification of the Howgrave-Graham algorithms, all the random bits in π_β are used, for simplicity's sake. The resulting algorithms implement these functions: in HG-1, $e = \pi_\beta(K\!:\!p\!+\!r)$ and in HG-2, $e = \pi_\beta(Kp\!+\!r)$. In both cases, $|k| + |r| = n^{3/4}$ random bits are random seeds and $|p| = n^{1/2}$ bits are fixed in π_β.

Because of the assumptions needed for π_β to produce n pseudo-random bits from $n^{3/4}$ random ones, π_β cannot be extremely simple. Unless a simpler instantiation can be found, a π_β based upon a standard computational assumption needs to be used.

Compared to a simpler algorithm that is developed in Chapter 7, denoted (G4, A4), the Howgrave-Graham algorithm involves an additional computational assumption and likely increases the time complexity of the backdoor generation algorithm. This comparison is detailed in Subsection 7.2.2.

Simplicity. Both algorithms have a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because both algorithms consist of almost the same steps as the honest algorithm.

confidentiality			yes: one-way π_β	good
completeness			yes: Coppersmith corollaries	good
indisting.	asymmetry		no: symmetric key β	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{1}{2}k}$	good
		$\mathcal{C}_{G_1}$	0, pseudo-random π_β	average
		$\mathcal{V}_{G_1}$	0	**failed**
		$\mathcal{H}_{G_1}$	1, pseudo-random π_β	average
	GKR		no: p and e are dependent	**failed**
	side channels	complexity	$t_n(G_1) = t_n$ $t_e(G_1) \approx T(\pi_\beta) \cdot t_e$	good / F-rel.
		memory	no	good
computational assumptions			one way, pseudo-random π_β	average
simplicity			LS	good

Table 6–2: Properties of Algorithm HG-1 of Figure 6–2 and of of Algorithm HG-2 of Figure 6–3. Our trivial improvement, in the form of the use of π_β, is accounted for.

6.2 Kleptography: Young-Yung algorithms

There are five papers on kleptography by Young and Yung that relate to backdoors [YY96, YY97a, YY97b, YY05a, YY05b]. No algorithm from [YY97b] is listed because it establishes backdoors on types of cryptographic algorithms other than key generators, that is, on algorithms for signature, encryption, and key exchange.

6.2.1 YY96

The following first five RSA backdoored key generators are from [YY96]. The idea of the first two algorithms is to use yet another RSA key to encrypt the private parameter, p, into the public one, (n, e). The last three algorithms are the application of similar ideas to the design of EG backdoored key generators. (The honest EG key generator is given in Figure 7–11.)

First algorithm

The algorithm is given in Figure 6–4. It is the simplest asymmetric backdoored key generator. Denote the designer's public key by (N, E) and its private key by D, with $|N| = k$. In other words, the designer's key, N, is half the length of the distinguisher's key, n. The backdoor information used to retrieve the user private key is p itself. It is encrypted as $p^E \bmod N$ into the public parameter e.

Algorithm properties. The algorithm encrypts p within e. It uses the lengths

$$\begin{aligned} k &= |p| = |q| \\ k &= |N| = |e| \end{aligned}$$

PARAMETERS:
• The designer's RSA public key is (N, E) and its private key is D, with $|N| = k$.

```
Algorithm YY96-RSA [Key gen]
1: Pick rp $q$ of size $k$.
2: repeat
3:     Pick rp $p$ of size $k$.
4:     Set $n = pq$.
5:     Set $e \equiv p^E \bmod N$.
6: until $\gcd(e, \phi(n)) = 1$.
7: Compute $d \equiv e^{-1} \bmod \phi(n)$.
8: return $(p, q, d, e)$.
```

```
Algo. A-YY96-RSA [Key retr.]
1: Input of $(n, e)$.
2: Set $p \equiv e^D \bmod N$.
3: Compute $d \equiv e^{-1} \bmod \phi(n)$.
4: return $d$.
```

Figure 6–4: Young and Yung (1996): an asymmetric backdoor for RSA that encrypts p within e.

Therefore, the designer key is half the size of the user's. Also, e is constrained to be half the size that it may be in general. Nevertheless, this is not necessarily a weakness, because smaller e's are often used in practice [5] .

Confidentiality. Given the secure key lengths listed in Section A.1, either the choice of N makes the backdoor unconfidential or forces the choice of a larger n. This is contradictory to Definition 4.2.9.

Completeness. The backdoor information is encrypted with RSA itself. Therefore, completeness holds because the RSA function is invertible.

Symmetry. This backdoored key generator is asymmetric because the backdoor information is encrypted with RSA itself and RSA is asymmetric.

[5] A common industry-standard choice for e is $65537 = 2^{16}+1$, which has low Hamming weight and facilitates fast exponentiation, as mentioned in Subsection 2.2.8. In the theoretical RSA cryptosystem, e is bounded above by $\phi(n) \approx 2^{1024}$, for RSA-1024.

Cardinality. There are about $2^{k-\lg k}$ choices of parameter e, which is the number of generable parameters p, by Example 2.2.7.

$$\#\{e\} = \#\{p\}.$$

The cardinality of KSM is counted per value of n, hence per value of p. This is because the backdoor information, p, is embedded in e. In the honest key generation algorithm, e is generated given n: that $e \in \mathbb{Z}^*_{\phi(n)}$ makes it dependent on n. Still in the honest case, there remains close to $2k$ bits of freedom in the choice of e. However, for an already fixed p, setting $e \equiv p^E \bmod N$ in Step 5 leaves only one possibility for e. Therefore for KSM, $\mathcal{N}_{G_1,e} = 1$ while for KS, $\mathcal{N}_{G_0,e} \approx 2^{2k}$.

Computing the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms are the same so can be canceled. This is because G_0 and G_1 generate n in the same manner and $\mathcal{N}_{G_1,n} = \mathcal{N}_{G_0,n}$. The following ratio is yielded.

$$\mathcal{R}_{G_1} = \frac{\cancel{\mathcal{N}_{G_1,n}}}{\cancel{\mathcal{N}_{G_0,n}}} \frac{1}{\mathcal{N}_{G_0,e}} \approx 2^{-2k}.$$

Modification. Nevertheless, this can be trivially improved and we do so for the analysis in order to compare algorithm design strategies as significantly as possible. Corollary 6.1.8 allows the factorization of n from the knowledge of only half the bits of p. Therefore, only this half, which is $k/2$-bit long, needs to be used as backdoor information. The $k/2$ remaining bits to be embedded in e can be random according to the algorithm in Subsection 2.2.8. The embedded backdoor information is half of p appended to some randomness, r. In the trivially modified G_1, Step 5 is hence replaced by

$$e = (p\rceil^{k/2} \mathbin{:} r)^E \bmod N,$$

for $r \in_U \{0,1\}^{k/2}$.

Therefore, with this trivial modification of the algorithm, there are about $k/2$ values of e per p, and

$$\begin{aligned}\mathcal{R}_{G_1} &= \frac{\cancel{\mathcal{N}_{G_1,n}}}{\cancel{\mathcal{N}_{G_0,n}}} \frac{\#\{e\}}{\mathcal{N}_{G_0,e}} \approx \frac{\#\{r\}}{2^{2k}} = \frac{2^{k/2}}{2^{2k}} \\ &= 2^{-\frac{3}{2}k}.\end{aligned}$$

Distribution properties. For the assumption on the approximate uniformity of the distribution of the RSA encryption function to hold as in Subsection 2.2.8, Proposition 2.2.27, a portion of the encrypted value must be random. As for the preceding property of *Cardinality*, Corollary 6.1.8 allows the factorization of n from the knowledge of only half the bits of p. Therefore, with this trivial modification of the algorithm, $\mathcal{C}_{G_1} = 0$.

The uniformity correctness, $\mathcal{C}_{G_1}$, is "good", if the RSA encryption exponent is uniform and the RSA encryption function's output's distribution is close to uniform (Subsection 2.2.8). Therefore, they are not computationally distinguishable from one another unless e is of a special form or the assumption on the uniformity of RSA fails.

Overall, this property rests on the assumption of the uniformity of the RSA encryption function (the function on which the assumption is made is explicitly provided).

Note. That the embedding is in e rather than in n makes appending random bits easier. It is easier to append (assumedly) uniformly distributed bits and this parameter is usually (in honest key generation) distributed uniformly, unless (obviously) e is restricted to be of a special form.

Entropy of embedding. If p is used entirely as backdoor information, then the entropy is null. In GKR, if p is fixed, n is fixed as well, thus a predictable situation occurs.

As in the modification introduced in *Cardinality*, if the embedded backdoor information is half of p and some randomness, r, then

$$(p\rceil^{k/2}\!:\!r)^E \bmod N$$

is close to uniform (Proposition 2.2.27). The encryption of the information corresponding to $p\rceil^{k/2}$ is likewise distributed.

Generalized key regeneration and variable correlation. The values of p and e are linked so that generalized key regeneration is not possible. Because e is computed from p, their values vary accordingly together. Unless specific collisions in RSA can be found, the values of e and p cannot be regenerated independently. Therefore, the security of RSA is mutually exclusive with the generalized key regeneration of this algorithm.

The modification introduced in *Cardinality* affects GKR, but does not change the overall result: there is at least one type of regeneration that is impossible. On one hand, about $\#\{r\} = n/4$ values of e are generable, per p. Therefore, if the legitimate user fixes p and asks to regenerate e, it is straightforwardly feasible. However, the inverse which consists in fixing e and regenerating p is infeasible. This regeneration would involve finding p', r' such that

$$(p'\rceil^{k/2} : r')^E \bmod N = (p\rceil^{k/2} : r)^E \bmod N.$$

Therefore, it would require the inversion of the RSA function, which is one-way (Subsection 2.2.2).

Complexity. The complexity is not analyzed for individual parameters, n and e, but for the whole of G_1. This is for the same reason that implies that GKR does not apply for this algorithm. The key generation steps are structured very differently from G_0, as the generation of p is embedded in the loop of the generation of e, and also involves an additional call to RSA-k.

Consider the generation of e, that is, the loop from Steps 2 to 6. The number of execution of this loop is about t_e. The complexity of one execution is about t_p added to $T(\text{RSA-}k)$. Therefore, the complexity of this algorithm is

$$\begin{aligned} T(G_1) &\approx t_q + t_e\,(t_p + T(\text{RSA-}k)) \\ &\approx t_n + t_e\,(t_n + T(\text{RSA-}k)) \\ &> t_e\,t_n \\ &> t_n + t_e = T(G_0). \end{aligned}$$

The complexity is rated “poor” because it is more than linear but less than quadratic in the complexity of G_0.

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses no additional computational assumption.

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			no: Def. 4.2.9 unsatisfied	**failed**
completeness			yes: RSA fct. is invertible	good
indisting.	asymmetry		yes: RSA is asymmetric	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{3}{2}k}$	**failed**
		$\mathcal{C}_{G_1}$	0, given RSA (Proposition 2.2.27)	average
		$\mathcal{V}_{G_1}$	0	**failed**
		$\mathcal{H}_{G_1}$	1, given RSA (Proposition 2.2.27)	average
	GKR		no: p and e are dependent	**failed**
	side-channels	complexity	$T(G_1) = t_n + t_e\,(t_n + T(\text{RSA-}k))$	poor
		memory	no	good
computational assumptions			none added	good
simplicity			LS	good

Table 6–3: Properties of Algorithm YY96-RSA of Figure 6–4. Our trivial improvement, as detailed in *Cardinality*, is accounted for.

Second algorithm

The second RSA backdoored key generator encrypts p within n, and is called PAP, for *pretty awful privacy* [6] .

Parameters:
• The designer's RSA public key is (N, E) and its private key is D, with $|N| = k$.
• Fixed key K, and $F_K : \{0,1\}^k \rightarrow \{0,1\}^k$, $G_K : \{0,1\}^k \rightarrow \{0,1\}^k$ be invertible pseudo-random functions: F_K is used to insure $\rho < N$ and G_K, "for pseudo-randomness" [7] .

Algorithm PAP [RSA key gen]

```
1: Pick **rp** $p$ of size $k$.
2: Set $\rho_1 = F_K(p)$ (increment $K$ until $p < N$).
3: Set $\rho_2 \equiv {\rho_1}^E \bmod N$.
4: Set $\rho = G_K(\rho_2)$ (increment $K$ until $p < N$).
5: repeat
6:     Set $n = \rho : r$, for a random $r$ of size $k$.
7: until $q = \lfloor n/p \rfloor$ is prime.
8: Set $n = pq$ and $e = 15$ (the initial $e$ will be 17).
9: repeat
10:    Set $e = e + 2$.
11: until $\gcd(e, \phi(n)) = 1$.
12: Compute $d \equiv e^{-1} \bmod \phi(n)$.
13: return $(p, q, d, e)$.
```

Algorithm A-PAP [Key retrieval]

```
1: Input of $(n, e)$.
2: Set $\rho : r = n$, such that $|\rho| = |r| = k$.
3: repeat
4:     Set $\rho = G_K^{-1}(\rho)$.
5:     Set $\rho \equiv \rho^D \bmod N$.
6:     Set $p = F_K^{-1}(\rho)$.
7: until $q = \lfloor n/p \rfloor \in \mathbb{Z}$ (incr. the $K$'s indep. o.w.).
8: Compute $d \equiv e^{-1} \bmod \phi(n)$.
9: return $d$.
```

Figure 6–5: Young and Yung (1996): PAP is an asymmetric backdoor for RSA that encrypts p within n.

[6] PAP is named to contrast with PGP, for *Pretty Good Privacy* [PGP06].

[7] The precise statement in [YY96, p.5] is "to achieve a pseudo-randomness of the values". This is meant in the sense of achieving a good distribution, not a theoretically pseudo-random one, since the key, K, is fixed. As detailed in *Distribution properties*, the modification explained in *Cardinality* suffices to argue pseudo-randomness.

Algorithm properties. Let $|p| = |q| = k = |N|$. Let F_K, G_K be invertible pseudo-random functions keyed with a fixed K. The role of F_K is to insure that p is less than N and G_K is for pseudo-randomness.

Confidentiality. Given the secure key lengths listed in Section A.1, either the choice of N makes the backdoor unconfidential or forces the choice of a larger n. This is contradictory to Definition 4.2.9.

Completeness. The backdoor information is encrypted with RSA itself. Therefore, completeness holds because the RSA function is invertible.

Symmetry. This backdoored key generator is asymmetric because the backdoor information is encrypted with RSA itself and RSA is asymmetric.

Cardinality. This computation was done with more intuition as Example 5.1.5, because it aims at giving intuition for all such computations. A more general version of the following computations is given in the context of Subsection 7.4.1, and finalized as Equation 7.24.

Then p divides $\rho\!:\!r$ with probability given by the fraction of strings covered by multiples of p times the expected fraction of strings covered by $\rho\!:\!r$. This is:

$$\begin{aligned} \Pr\left[\frac{\rho\!:\!r}{p} \in \mathbb{Z}\right] &= \frac{\#\{2k\text{-bit multiples of } p\}}{2^{2k}} \cdot \frac{\#\{\rho\!:\!r\}}{2^{2k}} \\ &= \frac{2^k}{2^{2k}} \cdot \frac{2^k}{2^{2k}} = \frac{1}{2^{2k}}. \end{aligned}$$

Nevertheless, substituting r by an appropriate $r' = \lfloor \frac{\rho:r}{p} \rfloor \cdot p - (\rho : 0^k)$ allows for the exact division by p.

$$\Pr\left[\frac{\rho\!:\!r'}{p} \in \mathbb{Z}\right] = 1$$

By Theorem 2.2.4, a k-bit integer is prime with probability of about $2^{-\lg k}$. Therefore

$$\Pr\left[\frac{\rho : r'}{p} \text{ is prime}\right] \;=\; 1 \cdot 2^{-\lg k} = \frac{1}{2^{\lg k}}.$$

The remaining part of the analysis is

$$\begin{aligned}
\mathcal{N}_{G_1,n} &= \#\{\rho : r' \mid \exists \text{ prime } q \text{ s.t. } pq = \rho : r'\} \\
&= \#\{p\} \cdot \#\{r'\} \cdot \Pr\left[\frac{\rho : r'}{p} \text{ is prime}\right] \\
&\approx 2^{k-\lg k} \cdot 1 \cdot 2^{-\lg k} \\
&= 2^{k-2\lg k}. \qquad (6.1)
\end{aligned}$$

In the ratio of the cardinalities of KSM and KS, the $\#\{e\}$ terms can be canceled so that

$$\mathcal{R}_{G_1} \;=\; \frac{\mathcal{N}_{G_1,n}}{\mathcal{N}_{G_0,n}} \approx \frac{2^{k-2\lg k}}{2^{2k-2\lg k}} = 2^{-k}.$$

Modification. On the other hand, it may seem, but is not the case, that using the same trivial modification as for Algorithm YY-96-RSA (Figure 6–4), a greater cardinality is achievable. Suppose that ρ' be the encryption of the upper half of the bits of p appended with $|p|/2$ random bits. This means that Step 3 of G_1 is replaced by

$$\text{Set } \rho'_2 \equiv \left(\rho_1\rceil^{|p|/2} : r\right)^E \bmod N,$$

for $r \in_U \{0,1\}^{|p|/2}$. Then Step 4 computes ρ' with ρ'_2.

Although ρ' is less redundant than ρ, this change has no effect on cardinality. This is because p is embedded in n and not in e, so there is no need for extra randomization: fixed chosen bits in one parameter do not affect the other. Nevertheless, this trivial modification is beneficial for other algorithms properties, such as *Distribution properties.* Thus it is retained.

Distribution properties. An advantage of this algorithm is that the selection of e can be biased toward the smaller ones, so e is typically small, as in PGP. Therefore, with respect to PGP, the e part of the public key has good indistinguishability.

For the n part, note again that via the trivial modification (detailed in *Cardinality*), the encryption of the backdoor information is close to uniformly distributed (Proposition 2.2.27). Despite this, indistinguishability does not holds, as pointed out in [CS03, Section 5]. In the setting of this work, this is a case where uniformity correctness (Definition 5.1.10) was not taken into account. The idea is that the upper bits of n are not uniformly distributed. If two random 512-bit primes are picked, their product is 1023-bit long with probability 38%. With probability 48% it is 1024-bit long with leading bits 10 and with probability 14% it is 1024-bit long with leading bits 11. Nevertheless, this problem does not happen if p and q are picked over the more appropriate interval $[\sqrt{2} \times 2^{511}, 2^{512} - 1]$.

Entropy of embedding. Again via the trivial modification (detailed in *Cardinality*), the encryption of the backdoor information is close to uniformly distributed (Proposition 2.2.27). The encryption of the information corresponding to $p\rceil^{|p|/2}$ is close to uniformly distributed in the upper half of the bits of n, accounting for an entropy between 0 and 1.

Generalized key regeneration and variable correlation. The modification introduced in *Cardinality* affects GKR, but does not change the overall result: there is at least one type of regeneration that is impossible. On one hand, about $p/2$ values of ρ are generable, per p. Therefore, if the legitimate user fixes p and asks to regenerate q, it is straightforwardly feasible.

However, the inverse which consists in fixing q and regenerating p is infeasible. Let s be $|p|/2$ random bits. Set

$$\rho_{p,s} = G_K\left(F_K(p\rceil^{|p|/2} : s)^E \bmod N\right).$$

This regeneration would involve finding p', r', s' such that

$$\rho_{p',s'} : r' = \rho_{p,s} : r.$$

Therefore, it would require the inversion of the RSA function, which is one-way (Subsection 2.2.2).

Complexity. It seems that there could be an efficiency issue with Steps 5 to 7, in parallel to the preceding one detailed in *Cardinality*. Nevertheless,

$$q = (\rho : r)/p$$

is a pseudo-random integer and q is prime with probability of about $1/2^{-\lg k}$ (Equation 6.1). Therefore, the complexity of generating q in G_1 is the same as in G_0 and the one of generating n is:

$$\begin{aligned} t_n(G_1) &= t_p + T(F_K) + T(\textsc{rsa}\text{-}k) + T(G_K) + t_q \\ &= t_n + T(F_K) + T(G_K) + T(\textsc{rsa}\text{-}k). \end{aligned}$$

Also,

$$t_e(G_1) = t_e,$$

assuming that incrementing e in Step 10 has about the same expected complexity as picking a random e.

This means that if $T(F_K)$ and $T(G_K)$ have at most a complexity similar to the one of $t_n + T(\textsc{rsa}\text{-}k)$, then G_1's complexity is rated as "good". Firstly, it is linear in the one of G_0. Secondly, RSA-k has a complexity negligible w.r.t. the one of RSA-$2k$, of which t_n makes the most part. Therefore, RSA-k has a complexity negligible w.r.t. t_n. Overall, G_1 is rated as "good / F-relative".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses pseudo-random functions, F_K and G_K, which are not explicitly provided.

Simplicity. The algorithm features a more complicated prime generation than G_0. Compared with other algorithms that follow, this makes the algorithm only a little more complicated than G_0.

confidentiality			no: Def. 4.2.9 unsatisfied	**failed**
completeness			yes: RSA fct. is invertible	good
indisting.	asymmetry		yes: RSA is asymmetric	
	diversity	$\mathcal{R}_{G_1}$	2^{-k}	poor
		$\mathcal{C}_{G_1}$	> 0 , bad distribution of n	poor
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, given RSA	average
	GKR		no	**failed**
	side channels	complexity	$t_n(G_1) = t_n + T(F_K) + T(\text{RSA-}k) + T(G_K)$ $t_e(G_1) = t_e$	good / F-rel.
		memory	no	good
computational assumptions			pseudo-random F_K and G_K (used with the ones of Proposition 2.2.27)	average
simplicity			LS-MS	average

Table 6–4: Properties of Algorithm PAP of Figure 6–5. Our trivial improvement, as detailed in *Cardinality*, is accounted for.

Third and fourth algorithms

The last three algorithms are EG backdoored key generators. The following two algorithms (the third and fourth algorithm of Young and Yung's 1996 paper) however use RSA to encrypt the backdoor information.

PARAMETERS:
• The designer's RSA public key is (N, E) and its private key is D, with $|N| \geq |a|$.
• The parameter p can be kept fixed, but assume that it is picked randomly.

Algorithm YY96-EG-p [Key gen]

```
1: repeat
2:     Pick random $0 \leq a \leq p-2$.
3:     Set $\alpha \equiv a^E \bmod N$.
4: until $\alpha < p$ and is a generator mod$p$.
5: Compute $\beta \equiv \alpha^a \bmod p$.
6: return $(p, \alpha, a, \beta)$.
```

Algo. A-YY96-EG-p [Key retr.]

```
1: Input of $(\alpha, \beta, p)$.
2: Set $a \equiv \alpha^D \bmod N$.
3: return $a$.
```

Figure 6–6: Young and Yung (1996): an asymmetric backdoor for EG which embeds a in α and keeps p fixed.

PARAMETERS:
• The designer's RSA public key is (N, E) and its private key is D, with $|N| \geq |a|$.
• The parameter α can be kept fixed, but assume that it is picked randomly in $\mathbb{Z}_p^*$.

Algorithm YY96-EG-α [Key gen]

```
1: repeat
2:     Pick random $0 \leq a \leq p-2$.
3:     Set $p \equiv a^E \bmod N$.
4: until $\alpha < p$ and is a generator mod$p$.
5: Compute $\beta \equiv \alpha^a \bmod p$.
6: return $(p, \alpha, a, \beta)$.
```

Algo. A-YY96-EG-α [Key retr.]

```
1: Input of $(\alpha, \beta, p)$.
2: Set $a \equiv p^D \bmod N$.
3: return $a$.
```

Figure 6–7: Young and Yung (1996): an asymmetric backdoor for EG which embeds a in p and keeps α fixed.

Third and fourth algorithms' properties. The first two of these three EG backdoored key generators are very similar, as to justify analyzing them together. The first one, given in Figure 6–6, keeps p fixed. This is useful when p is shared amongst many users. The second EG backdoored key generator, given in Figure 6–7, keeps α fixed. The difference is that p plays the role that α did, in the previous algorithm.

Confidentiality. Given the secure key lengths listed in Appendix A.1, either the choice of N makes the backdoor unconfidential or forces the choice of a larger n. The key lengths are contradictory to Definition 4.2.9 because $|\alpha| = |N|$. Keys for EG are smaller than for RSA, if only by a small fraction.

Completeness. The backdoor information is encrypted with RSA. Therefore, completeness holds because the RSA function is invertible.

Symmetry. This backdoored key generator is asymmetric because the backdoor information is encrypted with RSA, which is asymmetric.

Cardinality. There are $\mathcal{N}_{G_0} \approx 2^{3|p|}$ honest EG keys (Appendix A.2.1). In Step 3 of YY96-EG-p or -α, since the RSA function is a permutation and its input, a, covers all the values smaller than $p-1$, all the α's are generable. There are $p-1$ generable a's, each yielding one value of α and, in turn, of β. This accounts to about $2^{|p|}$ generable backdoored keys. In the ratio of the cardinalities of KSM and KS, the $\#\{a\}$ terms can be canceled, one of p or α is fixed and the other is a permutation of a, so that $\mathcal{R}_{G_1} \approx 2^{-2|p|}$.

Distribution properties. It is not possible to achieve a good distribution for the embedded backdoor. If using the same trivial modification via Proposition 2.2.27

as for Algorithm YY-96-RSA (Figure 6–4) was possible, then the distribution of the encrypted backdoor information would be indistinguishable from a uniformly distributed α or p. However, all the bits of either of these parameters are necessary to store the value of the backdoor information, a. In the worst case where a is kept fixed by request of the distinguisher, $\mathcal{C}_{G_1} = 1$.

Entropy of embedding. In order to use the approximate uniform distribution of the RSA encryption (Proposition 2.2.27), one would need to pad a with some random bits. However, the lengths of α and a are about the same, so there is no place to do so.

If a is used entirely as backdoor information, then the entropy is null. In GKR, if a is fixed, α (or p) is fixed as well, thus a predictable situation occurs.

Generalized key regeneration and variable correlation. This property is not achievable because a are either α or p are dependent.

Complexity. The RSA encryption is performed inside the loop that checks whether α is a generator modulo p. Therefore, the relative complexity of G_1 is t_α times the one of RSA on $|p|$ bits.

Because either p or α is fixed, the total complexity does not involve a $T(G_0)$ term. The total complexity in the following. In the first case:

$$t_\alpha \cdot T(\text{RSA-}|p|)$$

or, in the second case:

$$t_p \cdot T(\text{RSA-}|p|).$$

Overall, the complexity is rated as “poor”. It is less than quadratic in the complexity of G_0 times the one RSA-$|p|$, which is negligible w.r.t. t_α or t_p.

Memory. No use of additional memory is made.

Computational assumptions. The RSA cryptosystem is used for encrypting the backdoor information.

Simplicity. Despite their high computational complexity, these algorithms are very simple, as they mimic the operations of G^{EG}, except for the obvious additional encryption.

confidentiality			no: Def. 4.2.9 unsatisfied	failed
completeness			yes: RSA fct. is invertible	good
indisting.	asymmetry		yes: RSA is asymmetric	
	diversity	$\mathcal{R}_{G_1}$	$2^{-2\|p\|}$	failed
		$\mathcal{C}_{G_1}$	1	failed
		$\mathcal{V}_{G_1}$	0	failed
		$\mathcal{H}_{G_1}$	0	failed
	GKR		no	failed
	side-channels	complexity	$t_\alpha \cdot T(\text{RSA-}\|p\|)$ or $t_p \cdot T(\text{RSA-}\|p\|)$	poor
		memory	no	good
computational assumptions			RSA	average
simplicity			LS	good

Table 6–5: Properties of the related algorithms of Figures 6–6 and 6–7. Both are backdoored key generators for EG which keep one of the keys’ parameters fixed.

Fifth algorithm

The third EG backdoored key generator is pure EG, because it uses ElGamal to encrypt the backdoor as well. The idea is to encrypt the private key, a, as the backdoor value, into the public parameter p. This encryption is done with EG.

PARAMETERS:
• The designer's EG public key is $(\bar{p}, \bar{\alpha}, \bar{\beta})$ and its private key is $\bar{a}$. It holds that $\bar{\beta} \equiv \bar{\alpha}^{\bar{a}} \bmod \bar{p}$, with $|\bar{p}| = |p|$.
• For a "private" (secret) key K, and let $F_K : \{0,1\}^{|p|} \rightarrow \{0,1\}^{|p|}$, be an invertible pseudo-random function, used for randomization [8] .

```
Algorithm YY96-EG-pure [Key gen]
 1: Input of $(K, \bar{p}, \bar{\alpha}, \bar{\beta})$.
 2: Pick random $0 \leq a \leq p - 2$.
 3: repeat
 4:     Pick random $\bar{k} \in \mathbb{Z}^*_{\bar{p}-1}$.
 5:     Set $p_1 \equiv \bar{\beta}^{\bar{k}} a \bmod \bar{p}$ (this encrypts $a$).
 6:     Set $p \equiv F_K(p_1)$.
 7:     if $p$ is prime and $a < p$
 8:         Set $\alpha \equiv \bar{\alpha}^{\bar{k}} \bmod \bar{p}$.
 9: until $\alpha < p$ and is a generator mod$p$.
10: Compute $\beta \equiv \alpha^a \bmod p$.
11: return $(p, \alpha, a, \beta)$.
```

```
Algo A-YY96-EG-pure [Key retr.]
1: Input of $(K, \alpha, p, \bar{a}, \bar{p})$.
2: Set $a \equiv F_K^{-1}(p)/\alpha^{\bar{a}} \bmod \bar{p}$ (this decrypts $a$).
3: return $a$.
```

Figure 6–8: Young and Yung (1996): a symmetric "pure EG" backdoor for EG which embeds a in p.

Algorithm properties. The keyed pseudo-random function, F_K, is used for randomization in the following way. The key K is incremented until Step 6 succeeds so that a prime p is generated out of the seed p_1.

[8] The precise statement in [YY96, p.8] is that p_1 "is encrypted with a private key to create pseudo-random functions". This differs from PAP (Figure 6–5): a pseudo-random distribution is achieved because the key, K, is secret, as detailed in *Distribution properties.*

Confidentiality. Algorithm YY96-EG-pure does not share the secure key length problem of the first four algorithms. This property rests on the security of the El-Gamal encryption. It has been shown that the semantic security of the ElGamal encryption is equivalent to the decision Diffie-Hellman problem [TY98]. Overall, because EG keys are backdoored, no additional computational assumption is required.

Completeness. The backdoor information is encrypted with EG itself. Therefore, completeness holds because the EG function is invertible.

Symmetry. This backdoored key generator is symmetric because the backdoor information is encrypted with F_K, a pseudo-random function that is indexed with a secret key K.

Cardinality. To count the number of generable keys, consider the three normally independently generated parameters, (p, α, a). Firstly, a is picked at random in its normal range, yielding about p values. Secondly, p is the encryption of a, but there are $|\bar{k}| = |\phi(p-1)| \approx |p|$ bits of randomness used in generating it. The same random bits are used to compute α from the designer's key corresponding parameter, so p and α are approximately permutations of one another. Therefore, $2^{2|p|}$ backdoored keys are generable. In the ratio of the cardinalities of KSM and KS, the $\#\{a\}$ and $\#\{p\}$ terms can be canceled so that $\mathcal{R}_{G_1} \approx 2^{-|p|}$.

Distribution properties. Assuming that F_K is a pseudo-random function, a pseudo-random distribution is achieved for p because the key, K, is secret. This randomization of p involves of the use of symmetric cryptography.

The parameter p could also be argued to be properly distributed because of the semantic security of EG, used to encrypt a into it (refer to *Cardinality*). The parameter α is properly distributed because $\bar{k}$ being uniformly distributed implies the same for α. This would allow the backdoored generator to be asymmetric.

Entropy of embedding. From the pseudo-randomness of the EG encryption (Proposition A.3.6), $\mathcal{H}_{G_1} = 1$, given EG.

Generalized key regeneration and variable correlation. The values of p and a are linked so that generalized key regeneration is not possible.

Complexity. Just as for the preceding two algorithms, the encryption of the backdoor information (with EG instead of RSA in this case) is performed inside the α generation loop. The same applies to the application of F_K, which together with the EG encryption, generates p. Therefore, this α generation loop also includes a primality check for candidate values of p. Overall, the generation of a is the same as the honest one:

$$t_a(G_1) = t_a.$$

The generation of p and α are intertwined. There are two checks on the loop from Step 2 to 9, one for the primality of p, the other so that α is eventually a generator. Within the loop, there is an ElGamal encryption and a randomization via F_K.

$$t_{p,\alpha}(G_1) \quad = \quad t_\alpha \cdot t_p \cdot (T(\text{EG-}|p|) + T(F_K)).$$

Overall, the complexity is quadratic in the complexity of G_0. It is rated “failed”.

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a pseudo-random function, F_K, which is not explicitly provided.

Simplicity. Despite their high computational complexity, these algorithms are very simple, as they mimick the operations of G^{EG}, except for the obvious additional encryption. Since it is done via EG, it is probabilistic and is comparatively a little more complex than the one of the two preceding algorithms.

confidentiality			yes: EG	good
completeness			yes: EG function is invertible	good
indisting.	asymmetry		no: K is secret	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\|p\|}$	good
		$\mathcal{C}_{G_1}$	0, given F_K	average
		$\mathcal{V}_{G_1}$	0	**failed**
		$\mathcal{H}_{G_1}$	$\mathcal{H}_{G_1} = 1$, given EG	good
	GKR		no	**failed**
	side-channels	complexity	$t_a(G_1) = t_a$ $t_{p,\alpha}(G_1) = t_\alpha \cdot t_p \cdot (T(\text{EG-}\|p\|) + T(F_K))$	**failed**
		memory	no	good
computational assumptions			pseudo-random F_K	average
simplicity			LS-MS	average

Table 6–6: Properties of the Algorithm YY96-EG-pure of Figure 6–8.

6.2.2 YY97

The following RSA backdoored key generator is from [YY97a], and is called PAP-2, being the second version of PAP. It uses ElGamal to encrypt the backdoor: denote the designer's public key by $(\bar{p}, \bar{\alpha}, \bar{\beta})$ and its private key by $\bar{a}$, so that $\bar{\beta} \equiv \bar{\alpha}^{\bar{a}} \bmod \bar{p}$. The problems with the secure key length of the previous paper remain here, for Algorithm PAP-2.

In parallel, in this paper, Young and Yung present backdoored algorithms based on a primitive algorithm called a *(1,2)-leakage*. This type of algorithm usually requires the use of *non-volatile memory* (NM). Nevertheless, for backdoored key generation, the design of Algorithm PAP-2 manages to avoid this, even though it uses the principle of (1,2)-leakage. The use of NM is, in general, a significant drawback as in practice, the physical cryptographic device's memory can be reset anyway, and the stored values stored, lost. This is detailed in Subsection 5.2.2. Our algorithms avoid NM completely, which the only close exception being the use of *volatile* memory (VM), in order to allow generalized key regeneration for EG backdoored key generation. The use of volatile memory is not as significant of a drawback, as there is no requirement upon the resetting of the cryptographic device.

(1,2)-leakage. To clarify the functioning of the following Algorithm PAP-2, consider the primitive SETUP algorithm of Figure 6–9. Such a mechanism requires the generation of multiple contaminated values in order to transmit one value of some backdoor information. The transmission of two messages, m_1 and m_2, in order to recover one hidden parameter, c_2, is called a (1,2)-leakage. Known examples of

(1,2)-leakages recourse to non-volatile memory in order to store the parameter, c_1, used to generate the first message.

PARAMETERS:
- ElGamal is used to encrypt the backdoor: denote the designer's public key by $(\bar{p}, \bar{\alpha}, \bar{\beta})$ and its private key by $\bar{a}$, so that $\bar{\beta} \equiv \bar{\alpha}^{\bar{a}} \bmod \bar{p}$.
- The RSA security parameter equals the one of the DH key exchange: $|p| = |\bar{p}|$.
- W is odd and fixed and $t \in \{0, 1\}$ at random.
- a, b are not secret and are included in the device.
- $H : \mathbb{Z}_p \to \mathbb{Z}_{p-1}$ is a cryptographically strong hash function.

```
Algorithm DH-$m_1$ [First DH msg]
1: Pick a random $c_1 \in \mathbb{Z}_{p-1}$ and store it.
2: return $m_1 \equiv \bar{\alpha}^{c_1} \bmod \bar{p}$.
```

```
Algo. DH-$m_2(c_1)$ [Second DH msg]
1: Pick a random $t \in \{0, 1\}$.
2: Set $z \equiv \bar{\alpha}^{c_1 - Wt} \bar{\beta}^{-ac_1 - b} \bmod p$.
3: Set $c_2 = H(z)$.
4: return $m_2 \equiv \bar{\alpha}^{c_2} \bmod \bar{p}$.
```

```
Algo A-DH [Attack on discrete log]
1: Input of $(m_1, m_2, \bar{a})$.
2: Set $r \equiv m_1^a \bar{\alpha}^b \bmod p$.
3: Set $z_0 \equiv m_1 / r^{\bar{a}} \bmod p$.
4: if $m_2 = \bar{\alpha}^{H(z_0)} \bmod p$ (the case $t = 0$)
5:     return $H(z_0)$.
6: Set $z_1 \equiv z_0 / \bar{\alpha}^W \bmod p$.
7: if $m_2 = \bar{\alpha}^{H(z_1)} \bmod p$ (the case $t = 1$)
8:     return $H(z_1)$.
```

Figure 6–9: Young and Yung (1997a): an asymmetric backdoor for DH key exchange with (1,2)-leakage, i.e., transmission of two messages in order to transmit c_2.

The algorithm of Figure 6–9 is a SETUP on the Diffie-Hellman (DH) key exchange, i.e. an attack on the use of the discrete logarithm assumption, which allows the transmission of additional information. The two secret values of the exchange are a, b and the would-be corresponding honest messages are $\bar{\alpha}^a \bmod p, \bar{\alpha}^b \bmod p$. Note that the key honest process and the designer both use the same generator $\bar{\alpha}$.

Use of NM for backdoor recovery. As in Example 5.2.1, a first message allows the transmission of a function of an exponent, c_1. A second message transmits another function of this exponent, $f(c_1)$, such that given the two messages and a private key, $\bar{a}$, the designer can recover yet another function of the exponent, $g(c_1)$.

The designer needs to preserve the first message in non-volatile memory (NM). The second message and the private key alone are not sufficient to recover $g(c_1)$.

PARAMETERS:
• ElGamal is used to encrypt the backdoor: denote the designer's public key by $(\bar{p}, \bar{\alpha}, \bar{\beta})$ and its private key by $\bar{a}$, so that $\bar{\beta} \equiv \bar{\alpha}^{\bar{a}} \bmod \bar{p}$.
• The RSA security parameter equals the one of the DH key exchange: $|p| = |\bar{p}|$.
• W is odd and fixed and $t \in \{0, 1\}$ at random.
• a, b, K are not secret and are included in the device.
• $H : dom(H) \rightarrow [0, p-1]$ is a cryptographically strong hash function.
• $F_K : dom(F_K) \rightarrow [0, p-1]$ is a keyed pseudo-random function (for randomization).
LEFT OUT FOR SIMPLICITY:
• $PBRM : [0, R] \rightarrow [0, S]$ is an invertible function that preserves the uniform distribution of its domain and s.t. $S > R > S/2$. Here, $[0, R] = [0, \bar{p} - 1]$ and $[0, S]$ is successively $dom(H)$ and $dom(F)$.

Algorithm PAP-2 [RSA key gen]

```
1: Pick a random $c_1 \in \mathbb{Z}_{\bar{p}-1}$.
2: Set $z \equiv \bar{\alpha}^{c_1 - Wt}\bar{\beta}^{-ac_1 - b} \bmod \bar{p}$.
3: Set $p = H(z)$ and increment $p$ until prime.
4: Set $u = \bar{\alpha}^{c_1} \bmod \bar{p}$.
5: repeat
6:     Set $U = F_K(u)$.
7:     Solve for $q$ in $U{:}R = pq + r,\ R \in \{0,1\}^{|p|}$.
8:     Increment $K$.
9: until $q$ is prime.
10: Set $n = U{:}R - r$.
11: Pick a random $e \in \mathbb{Z}^*_{\phi(n)}$.
12: Compute $d \equiv e^{-1} \bmod \phi(n)$.
13: return $(p, q, d, e)$.
```

Algo. A-PAP-2 [Key retrieval]

```
1: Input of $(n, e)$.
2: Set $U{:}R - r = n$, such that $|U| = |p|$.
3: Set $u = F_K^{-1}(U)$.
4: Find $\bar{\alpha}^{c_1} \bmod \bar{p} = PBRM^{-1}(u)$.
5: Set $z \equiv \bar{\alpha}^{c_1}\ \bar{\alpha}^{-Wt}\bar{\beta}^{-ac_1 - b} \bmod \bar{p}$.
6: Set $z' = PBRM(z)$.
7: Set $p = H(z')$ and increment $p$ until prime.
8: Compute $d \equiv e^{-1} \bmod \phi(n)$.
9: return $d$.
```

Figure 6–10: Young and Yung (1997a): an asymmetric backdoor for RSA, PAP-2, via a DH key exchange. The information on p is embedded in n.

Algorithm properties. The backdoored key generator PAP-2 (Figure 6–10) is based on encrypting p in n via a DH key exchange. In the algorithm of Figure 6–9, the parameter c_2 that is transmitted corresponds to p in the one of Figure 6–10. Overall, PAP-2 internally performs a (1,2)-leakage via a DH key exchange.

Confidentiality. This property is meant to hold by noting that Step 5 of A-PAP-2 requires the knowledge of the private $\bar{a}$ in order to compute

$$\begin{aligned} z &\equiv \bar{\alpha}^{c_1}\ \bar{\alpha}^{-Wt}\bar{\beta}^{-ac_1-b} \bmod \bar{p} \\ &\equiv \bar{\alpha}^{c_1}\ \bar{\alpha}^{-Wt}\,(\bar{\alpha}^{c_1})^{(-a\bar{a})}\ \bar{\beta}^{-b} \bmod \bar{p} \end{aligned} \tag{6.2}$$

one needs knowledge of $\bar{a}$ unless EG or the DH key exchange is broken. Note that $\bar{\alpha}^{c_1} \bmod \bar{p}$ is known from Step 4. Aside from $\bar{a}$, the remaining variables are fixed or public parameters.

However, given the secure key lengths listed in Section A.1, the key lengths make the backdoor unconfidential or force the choice of a larger RSA parameter n, which is contradictory to Definition 4.2.9. For ElGamal DH to be secure, a key length of $|\bar{p}| \geq 997$ is necessary. For RSA, this key length is $2|p| \geq 1024$. However, the RSA security parameter equals the one of the DH key exchange: $|p| = |\bar{p}|$. Therefore, for PAP-2 to satisfy confidentiality, the generated backdoored RSA key length is of $2|p| \geq 1994 \approx 2048$. This is contradictory to the assumptions on E that follow Definition 4.2.9.

Completeness. The backdoor information is encrypted with a DH key exchange on which a (1,2)-leakage is based. A (1,2)-leakage involves the transmission of two messages through the parameter n, that are n itself and u, in order to transmit p. The DH key exchange is alike to the one of Figure 6–9 and is computed as explained for Equation 6.2. Therefore, completeness holds because of knowledge of $\bar{a}$ and of the leaked value of $\bar{\alpha}^{c_1} \bmod \bar{p}$.

Symmetry. This backdoored key generator is asymmetric because the backdoor information is encrypted with DH, which is asymmetric.

Cardinality. The number of generable keys is as follows. As its is the case for Algorithm PAP (Figure 6–5), Steps 5 to 9 of Algorithm PAP-2 are of the form

$$q = \frac{E(p\!:\!c_1) : R}{p}$$

where c_1 and R, both of the size of p, are random parameters. Summing up for all p's (as they are embedded in n), one expects

$$\mathcal{N}_{G_1,n} = \#\{n\} = 2^{|p|-2\lg|p|} = 2^{k-2\lg k}$$

backdoored keys. In the ratio of the cardinalities of KSM and KS, the $\#\{e\}$ terms can be canceled so that

$$\mathcal{R}_{G_1} = 2^{-k}.$$

Because p is encrypted in the form of its seed z' and not directly as the backdoor information, the same trivial modification as for Algorithm YY-96-RSA cannot be used. The encryption of the backdoor information cannot involve $|p|/2$ random bits instead of the lower half bits of p.

Distribution properties. At Step 10 of PAP-2, p is fixed, U pseudo-random and R random, so $q = (U\!:\!R - r)/p$ is a pseudo-random number of the appropriate length.

The same argument as for Algorithm PAP (Figure 6–5) holds. The parameter n is not properly distributed.

Entropy of embedding. From the pseudo-randomness of the F_K function the upper bits of n are pseudo-randomly distributed, but the same does not apply to the lower bits, as they must be adjusted so that q is prime. This yields the evaluation: $0 < \mathcal{H}_{G_1} < 1$, given F_K.

Generalized key regeneration and variable correlation. This property is satisfied similarly as for PAP with only minor modifications to the algorithms. The only difference when regenerating p is that an incremented key should be used as additional input to H. Then, in Step 7 of A-PAP-2, this key is incremented until a coherent p is generated.

Complexity. The complexity is comparable to the one of the PAP algorithm. The difference lies into that PAP-2's cryptographic functions, H and F_K, are applied within the generation loops of p and q instead of outside of these loops. This implies that the complexity of the cryptographic functions is multiplied with the one of the loop of the corresponding prime generation. Overall, for the generation of n:

$$t_p(G_1) \quad = \quad t_p \cdot T(H) + T(\textsc{dh})$$

and

$$t_q(G_1) = t_q \cdot T(F_K).$$

For the generation of e:

$$t_e(G_1) = t_e.$$

These complexities are linear in the complexity of G_0 and the use of DH has complexity comparable to RSA (the one of G_0), of which t_n is the dominant term. Assuming that $F = H$ is negligible with respect to t_n, Algorithm G_1 is rated "good".

Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_1 is rated "good / F-relative".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a cryptographic hash function, H, and a pseudo-random function, F_K, which are not explicitly provided. Furthermore, it uses the assumption that the DH key exchange is secure.

Simplicity. This algorithm is more complicated, compared to PAP, which was already rated as LS-MS. That a DH key exchange is set up is an additional level of structure w.r.t. the re-use of RSA encryption.

confidentiality			no: Def. 4.2.9 unsatisfied	**failed**
completeness			yes: DH exchange	good
indisting.	asymmetry		yes: DH is asymmetric	
	diversity	$\mathcal{R}_{G_1}$	2^{-k}	poor
		$\mathcal{C}_{G_1}$	> 0, bad distribution of n	poor
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, given F_K	poor
	GKR		yes	good
	side-channels	complexity	$t_p(G_0) = t_p \cdot T(H) + T(\text{DH})$ $t_q(G_1) = t_q \cdot T(F_K)$ $t_e(G_1) = t_e$	good / F-rel.
		memory	no	good
computational assumptions			pseudo-random H and F_K + DH	poor
simplicity			MS	average

Table 6–7: Properties of Algorithm PAP-2 of Figure 6–10. (Our trivial modification to PAP does not apply to PAP-2.)

6.2.3 YY05a

The following RSA backdoored key generator is from [YY05a]. Its name, PP, stands for Private Primes.

Indistinguishably of c. The prime p is generated through a keyed pseudo-random function, H, [9] with a seed, u. This seed is encrypted via Rabin into a parameter, c, which is stored in the public parameter, n. The following is to insure that c is distributed indistinguishably. One picks a random element c in $\mathbb{Z}_N^*$ which Jacobi symbol is distributed as for a random element x in $\mathbb{Z}_N^*$, but which also has Legendre symbol, with respect to each of P, Q, that is uniformly distributed between 1 and -1. For this purpose, let $e_0, e_1, e_2, e_3 \in \mathbb{Z}_N^*$ be four constants included in the generator such that

$$\begin{aligned} L(e_0/P) &= L(e_0/Q) = 1 \\ L(e_2/P) &= L(e_2/Q) = -1 \\ L(e_1/P) &= -1, \quad L(e_1/Q) = 1 \\ L(e_3/P) &= 1, \quad L(e_3/Q) = -1. \end{aligned}$$

Algorithm properties. The security is based on the encryption of the back-door information with the Rabin public-key cryptosystem. Its parameters are denoted as follows: the public key is N and the private key is (P, Q), such that $N = PQ$.

[9] The original version uses a *random oracle*, H, but for the practical purposes of a keyed pseudo-random function.

PARAMETERS:

- The designer's Rabin public key is N and its private key is (P, Q), s.t. $N = PQ$ and $|N| = k$.
- ID is a fixed parameter.
- $H_K : \{0,1\}^{k+|ID:i:j|} \to \{0,1\}^k$ is a keyed pseudo-random function, for a non-secret key K (in the original version, H is a random oracle).
- $e_0, e_1, e_2, e_3 \in \mathbb{Z}_N^*$ are four constants included in the generator such that $L(e_0/P) = L(e_0/Q) = 1$, $L(e_2/P) = L(e_2/Q) = -1$, $L(e_1/P) = -1, L(e_1/Q) = 1$, and $L(e_3/P) = 1, L(e_3/Q) = -1$.

Algorithm PP [RSA **key generation**]

```
1: Update $i = i + 1$ in non-volatile memory.
2: for $j = 0$ to $\infty$
3:    Pick random $x, u \in \mathbb{Z}_N^*$.
4:    Pick a random $b \in \{0, 1\}$.
5:    if $J(x/N) = 1$, set $c = e_0^b e_2^{b-1} u^2 \bmod N$.
6:    if $J(x/N) = -1$, set $c = e_1^b e_3^{b-1} u^2 \bmod N$.
7:    repeat
8:       $p = H_K(u:ID:i:j)$, next $|N|$-bit block.
9:    until $p > 2^{|N|-1}$ and is prime.
10:   Pick a random $R \in \{0,1\}^{|N|}$.
11:   Set $n' = c:R$.
12:   Solve for $q$ in $n' = pq + r$.
13:   if $q > 2^{|N|-1}$, $q < p$ is prime, and $|pq| \geq 2N$
14:      Set $n = pq$.
15:      Pick a random $e \in \mathbb{Z}_{\phi(n)}^*$.
16:      Compute $d \equiv e^{-1} \bmod \phi(n)$.
17:      return $(p, q, d, e)$.
```

Algorithm A-PP [Key retrieval]

```
1: Input of $(n, e, i)$.
2: Set $c:R = n$, s.t. $|c| = k$ (+poss. carry bit).
3: for $\ell = 0$ to 3
4:    Set $w = ce_\ell^{-1} \bmod N$.
5:    Set $\{u_0, u_1, u_2, u_3\} = \text{Rabin}^{-1}(w)$.
6:    for $b = 0$ to 3
7:       for each possible $j$ and block of $H_K$
8:          $p = H_K(u_b:ID:i:j)$, next $|N|$ bits.
9:          if $p|n$ and $p \neq 1, n$
10:            Compute $d \equiv e^{-1} \bmod \phi(n)$.
11:            return $d$.
```

Figure 6–11: Young and Yung (2005a): PP is an asymmetric backdoor for RSA using the Rabin cryptosystem. The information on p is embedded in n.

Confidentiality. The key lengths are: $|N| = k$. The designer's Rabin key N is half the length of the user's n. Given the secure key lengths listed in Section A.1, Definition 4.2.9 is unsatisfied.

Confidentiality does not hold, as opposed to what the proof of [YY05a, Section 4] suggests. This paper states that if confidentiality failed, then either $|N|$-bit composites, $N = PQ$, or $2k$-bit composites, $n = pq$, can easily be actored. Exactly because of the secure key lengths listed in Section A.1, $|N|$-bit composites can easily be factored. Therefore, the statement of [YY05a, Section 4] does not hold.

Completeness. The backdoor information is encrypted with the Rabin cryptosystem. Therefore, completeness holds because the Rabin encryption function is invertible.

Symmetry. In order to create a unique randomization for each public key generated by PP, one of the parameters in the seed from which the prime p is generated is kept in non-volatile memory, somewhat as a secret key. Even though the security of PP does not rest on it, this parameter, i, makes the algorithm symmetric.

Cardinality. The number of generable keys is accounted as the ones for Algorithms PAP and PAP-2.

Distribution properties. This property is the same as for Algorithms PAP and PAP-2.

Entropy of the embedding. From the pseudo-randomness of the H_K function the upper bits of n are pseudo-randomly distributed, but the same does not apply to the lower bits, as they must be adjusted so that q is prime. This yields the evaluation: $0 < \mathcal{H}_{G_1} < 1$, given H_K.

Generalized key regeneration and variable correlation. First suppose that H, the random oracle, is use instead of H_K, as it was the case in the original algorithm. If the user asks to keep p and regenerate the rest, there is no problem since p is generated from u. The regeneration algorithm merely makes sure that n still contains c, while finding another q that works.

However, if the user asks to keep q and regenerate p, the usual strategy would be to swap the roles of the primes, that is, to keep the same backdoor value but link q, instead of p, with u. However, that would mean

$$q = H(u) : \ell,$$

for fixed q and u, which is unlikely since H is a random oracle. Thus, GKR is not feasible.

Modification. If H_K is used instead, then as for Algorithms PAP and PAP-2, the value of K is increased in the regeneration of p. Thus, with this small modification, GKR is feasible.

Complexity. The complexity is similar to the one of Algorithm PAP-2, with DH replaced by Rabin, for Steps 3 to 6. The equivalent of Step 3 of PAP-2 are Steps 7 to 9, which adds a factor of the cost of H_K, for every iteration in the generation of p. The value of e is generated (Step 15) outside the loop for the generation of q (Steps 2 to 13), which otherwise, if the loops were embedded, would have made the complexity of this part of PP quadratic in the one of G_0. Overall, for the generation of n:

$$t_p(G_1) \quad = \quad T(\text{Rabin}) + t_p \cdot T(H_K)$$

and

$$t_q(G_1) = t_q$$

and for the generation of e,

$$t_e(G_1) = t_e.$$

These complexities are linear in the complexity of G_0 and the use of Rabin's cryptosystem has complexity comparable to RSA (the one of G_0), of which t_n is the dominant term. Assuming that $F = H_K$ is negligible with respect to t_n, Algorithm G_1 is rated "good". Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_1 is rated "good / F-relative".

The retrieval algorithm can be sped up, although this does not affect the relevant measure of complexity. This cryptosystem decrypts most ciphertexts to four possible plaintexts. The retrieval algorithm can be sped up by encrypting with the smallest ambivalent roots of the ciphertext. Before Step 7, test **if** $u > -u \bmod N$ and if so, set $u = -u \bmod N$.

Memory. This algorithm uses non-volatile memory. The stored variable i is a counter for the number of backdoored keys that the generator, G_1, created. It insures that the generator's pseudo-random seed is always different, thus the same keys are unlikely to be regenerated. The variable j may at first appear to already play this role of unique counter, but it is reset at zero every time the algorithm is run. Therefore, the uniqueness of the input to H_K comes only from i and u (but the latter, only with some probability).

This property was analyzed as Example 5.2.2.

Computational assumptions. The algorithm uses a keyed pseudo-random function, H_K, which is not explicitly provided. Furthermore, it uses the assumption that Rabin encryption is secure, which rests on integer factorization being difficult and which may be a stronger assumption than the one that RSA is secure (Subsection 2.2.7).

Simplicity. This algorithm is about as complicated as PAP-2, which was rated as MS.

confidentiality			no: Def. 4.2.9 unsatisfied	**failed**
completeness			yes: Rabin	good
indisting.	asymmetry		no: secret i	
	diversity	$\mathcal{R}_{G_1}$	2^{-k}	poor
		$\mathcal{C}_{G_1}$	> 0, bad distribution of n	poor
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, given H_K	poor
	GKR		yes	good
	side-channels	complexity	$t_p(G_0) = t_p \cdot T(H_K) + T(\text{Rabin})$ $t_q(G_1) = t_q$ $t_e(G_1) = t_e$	good / F-rel.
		memory	NM	average
computational assumptions			Rabin + pseudo-random H_K	poor
simplicity			MS	average

Table 6–8: Properties of Algorithm PP of Figure 6–11. Properties are given for our version, as described in *Generalized key regeneration and variable correlation*, with the random oracle H replaced by a family of pseudo-random functions, H_K.

6.2.4 YY05b

The following RSA backdoored key generator is from [YY05b]. As it uses point compression on elliptic curves, it is meant to be space efficient.

The original version of this backdoored key generator uses a random oracle, H, but for the practical purposes of a keyed pseudo-random function. In this context, H is sometimes applied to a given input so that it produces a long output of which consecutive blocks that can be selected until some desired properties are satisfied. Instead, we use a keyed pseudo-random function, H_K, and increase K until such properties are satisfied.

Algorithm properties. The designer encrypts the backdoor using elliptic curve Diffie-Hellman (DH) key exchange [10] . Two elliptic curves are used in order to save space. To simplify, consider the algorithm with only one curve E. Let G be a base point of order q on $E(\mathbb{F}_{2^m})$. The elliptic curve private key is picked as a random $x \in \mathbb{Z}_{q-1}$ and the corresponding public key is $Y = xG$. As usual, (G, Y) is included in the RSA key generator.

Consider Steps 2 and 3 of G_1 =EC-SETUP. Note that it holds that $s_{priv}, s_{pub} \in \{0,1\}^{m+1}$.

[10] For basic explanations on elliptic curves, see Appendix A.3.

Parameters:
• G is a base point of order q on $E(\mathbb{F}_{2^m})$.
• The designer's elliptic curve public key is $Y = xG$, for a random $x \in \mathbb{Z}_{q-1}$ which is its private key.
• $H_K : \{0,1\}^{m+1} \rightarrow \{0,1\}^k$ is a keyed pseudo-random function, where K is an initially fixed key.
• Θ is s.t. a carry bit in $n' - r$ (Step 13 in G_1) is unlikely.
• $\pi : \{0,1\}^\Theta \rightarrow \{0,1\}^\Theta$ is an efficiently invertible pseudo-random permutation.

Algorithm EC-SETUP(Y, G)
```
1: Pick a random $0 < k < q$.
2: Set $s_{priv} = kY$.
3: Set $s_{pub} = kG$.
4: Pick a random $e \in \mathbb{Z}_{2^{2k}}$.
5: repeat
6:    repeat
7:       Pick a random $\ell \in \{0,1\}^{k/2}$, increment $K$.
8:       Set $p = H_K(s_{priv}) : \ell$.
9:    until $p$ is prime, $|p| = k$, $p]^2 = 11$,
          $\gcd(p-1, e) = 1$.
10:   Pick a random $s \in \{0,1\}^{\Theta-(m+1)}$.
11:   Pick a random $R \in \{0,1\}^{2k-\Theta}$.
12:   Set $n' = \pi(s : s_{pub}) : R$.
13:   Solve for $q$ in $n' = pq + r$.
14: until $q$ is prime, $|q| = k$, $q]^2 = 11$,
          $\gcd(q-1, e) = 1$.
15: Set $n = pq$.
16: Compute $d \equiv e^{-1} \bmod \phi(n)$.
17: return $(p, q, d, e)$.
```

Algo. A-EC-SETUP(x)
```
1: Input of $(n, e)$.
2: Set $t : R = n$, s.t. $|t| = \Theta$ (+poss. carry bit).
3: Set $s : s_{pub} = \pi^{-1}(t)$, s.t. $|s_{pub}| = m + 1$.
4: Set $s_{priv} = x s_{pub}$.
5: repeat
6:    Set $u = H_K(s_{priv})$ and increment $K$.
7: until find $p$ w. poss. upper bits $u$ [Th. 6.1.8].
8: Compute $d \equiv e^{-1} \bmod \phi(n)$.
9: return $d$.
```

Figure 6–12: Young and Yung (2005b): an asymmetric backdoor for RSA, using the elliptic curve Diffie-Hellman (DH) key exchange. The information on p is embedded in n.

Confidentiality. To insure confidentiality, the prime p is generated through a pseudo-random function H_K with s_{priv} as seed which is stored via s_{pub} in the public parameter, n. Also, let π be an efficiently invertible pseudo-random permutation from Θ bits to Θ bits. The parameter Θ is chosen such that a carry bit in (the mid-upper bits of) $n' - r$ (Step 13 of Algorithm EC-SETUP) is unlikely. For this,

$|n'| \gg |r|$ is preferable, and in this case, it holds that $|n'| = 2k$ and $|r| = k - \Theta$ (same as the number of free bits for choosing the second prime q).

We deem it important to include the reasons why the proofs of Young and Yung, as in [YY05b], are incomplete. One of these proofs concern confidentiality [YY05b, Appendix A.2].

Critique of the confidentiality proof. The same critique applies as for *Indistinguishability*, which follows, as the provided proof is very similar.

Confidentiality proof (redone). The goal is to show that from the RSA backdoored public keys, it is computationally infeasible to find the corresponding private key. From the security of the cryptographic hash function π, one may have a hard time deducing s_{pub} from n, but π is not assumed to be one-way.

Given s_{pub}, it is infeasible to deduce s_{priv}, assuming the security of the designers elliptic curve public-key cryptosystem. Therefore, even with access to the keyed pseudo-random function, H_K, one cannot mimic Step 6 of Algorithm A-EC-SETUP in order to compute p.

Completeness. The DH key exchange has private parameter, s_{priv}, and public parameter, s_{pub}. These parameters are compressed forms of points on a given elliptic curve. To simplify, consider that $s_{priv} = kY$ and that $s_{pub} = kG$, for a random $0 < k < q$. Clearly, because the designer knows x and extracts $s_{pub} = kG$ from n, it can compute $s_{priv} = kY = x(kG)$. Because E is over $\mathbb{F}_{2^m}$, with compression, $s_{priv}, s_{pub} \in \{0,1\}^{m+1}$. Note that both s_{priv}, s_{pub} are included in the RSA key generator. This process insures completeness.

Symmetry. This generation algorithm uses only public parameters, Y, G, to generate the prime p. Confidentiality holds as x is kept secret. Therefore, x can be the same for all the distinguishers and uniformity holds. This backdoor is asymmetric.

Cardinality. Because p is embedded in n, we count the total number of n and e's. Firstly, consider the number of primes p is the number of primes of the form

$$p = H_K(s_{priv}) : \ell \tag{6.3}$$

for $\ell \in \{0,1\}^{k/2}$. By Example 2.2.7, the number of primes p is

$$\#\{p\} \approx \#\{\ell\} \cdot \frac{2^{k-\lg k}}{2^k} = 2^{k/2-\lg k}.$$

Secondly, consider the number of primes q is the number of primes of the form

$$q = \frac{E(p:s):R}{p} \tag{6.4}$$

where E is the generic encryption function (in practice, it is π here) and p stands for fixed information related to itself (in practice, it is s_{pub} here), and for $s \in \{0,1\}^{\Theta-(m+1)}$ and $R \in \{0,1\}^{2k-\Theta}$.

The parameter $m + 1 = |s_{pub}|$ is about the length of a compressed point on the elliptic curve over $\mathbb{F}_{2^m}$. Because the cardinality of the backdoored key set is expressed as a function of the security parameter of the cryptosystem which the key generator is from, one wants to express m as a function of $2k$. Let us assume that the ratios of the key lengths shown in Table A.1 hold as a general rule. Since RSA's $2k = 1024$ corresponds to EC's $|s| \approx 160$, for a compressed point $s \in E$. Then this assumption means that $m \approx \frac{1}{3}k$ by Definition 4.2.9.

Therefore,

$$m \approx 2q \approx \frac{2}{3}k.$$

Therefore, the total number of random bits in Equation (6.4) is

$$|s| + |R| \approx 2k - m \approx \frac{4}{3}k.$$

The number of q that pass the divisibility and primality tests is as for Algorithms PAP, PAP-2 and PP, so that the probability of q being an integer and being prime is about $1/(2^{k+\lg k})$. To sum up, the number of primes q is

$$\begin{aligned} \#\{q\} &\approx \frac{2^{|s|+|R|}}{2^{k+\lg k}} \\ &= 2^{k/3-\lg k}. \end{aligned}$$

Summing up the number of primes of the form of Equation (6.3) and Equation (6.4) as well as the number of possible e's, the total number of generable keys is

$$\begin{aligned} \mathcal{N}_{G_1,n} &= \#\{p\} \cdot \#\{q\} \\ &\approx 2^{k/2-\lg k} \cdot 2^{k/3-\lg k} \\ &= 2^{\frac{5}{6}k-2\lg k}. \end{aligned}$$

In the ratio of the cardinalities of KSM and KS, the $\#\{e\}$ terms can be canceled so that $\mathcal{R}_{G_1} \approx 2^{-\frac{7}{6}k}$.

Distribution properties. In the *Distribution properties* paragraph of PAP (Subsection 6.2.1), it is shown that n is not pseudo-random, unless its factors are chosen in a special range. This backdoored key generator chooses these factors in such a special range, but this makes both the primes and n distinguishable. For the backdoored and honest distributions to match, the backdoored one needs to be changed to match the honest one, not the contrary as it seems to be the intention here. Therefore the pseudo-randomness of the n portion of the backdoored public key is incorrect.

We deem it important to include the reasons why the proofs of Young and Yung, as in [YY05b], are incomplete. One of these proofs concern indistinguishability (idem, Appendix A.1) and confidentiality (idem, Appendix A.2).

Critique of the indistinguishability proof. Indistinguishability is proved as follows. Let $s = \pi^{-1}(n\rfloor_\Theta)$. Note that if n is backdoored, then $s = s_{pub}$. By contradiction, we suppose that a distinguisher D non-negligibly distinguishes primes backdoored with s_{pub}. The idea is to use D to invert the ECDDH key exchange, i.e. from (a_1G_0, a_2G_0), to find the shared secret $a_1a_2G_0$. Note that D makes calls to the keyed pseudo-random function (standing in for an oracle), H_K, which returns the shared secret with probability denoted p_{trap}.

1. First, we assume that p_{trap} is non-negligible. D is called on input primes computed from (a_1G_0, a_2G_0) as well as some random points. From this assumption on p_{trap}, one finds $a_1a_2G_0$ with non-negligible probability.
2. The complementary case is when p_{trap} is negligible, that is $p_{trap} < \gamma$, where γ is a negligible function in the security parameter. The reduction shown is claimed to succeed with probability proportional to $1 - \gamma$, so it appears that the reduction is successful when the keyed pseudo-random function, H_K, does not return the shared secret.

The problem is that there is then no explanation as to how the shared secret is computed from D. In fact, the output of the reduction is the value of D itself on parameters generated similarly as in the first case. Furthermore, on a more technical note, the output of D should be a yes or no, whether it has distinguished the primes as backdoored or not. A single bit cannot account for a ECDDH shared secret.

Therefore, the proof seems to miss all the difficult instances, that is, when the probability that the oracle gives the solution is negligible.

Entropy of the embedding. From the pseudo-randomness of the π function the upper bits of n are pseudo-randomly distributed, but the same does not apply to the lower bits, as they must be adjusted so that q is prime. This yields the evaluation: $0 < \mathcal{H}_{G_1} < 1$, given π.

Generalized key regeneration and variable correlation. Same as for PP in the preceding section, with s_{priv} and s_{pub} being equivalent to u and c respectively.

Modification. The same small modification as for PP applies as well.

Complexity. The complexity to the ones of Algorithms PAP, PAP-2 and PP. Accordingly, RSA or Rabin would be replaced by EC, but its associated costs are mostly offline.

Overall, for the generation of e, $t_e(G_1) = t_e$. For the generation of n, the generations of p and q are intertwined. The loop for the generation of q (Steps 5 to 14) includes the cryptographic function π (Step 9) as well as the loop for the generation of p (Steps 6 to 9). This loop in turn includes the function H_K.

$$\begin{aligned} t_n(G_1) &= t_q\left(T(\pi) + t_p \cdot T(H_K)\right) \\ &\approx t_n^2 \cdot T(H_K) + t_n \cdot T(\pi). \end{aligned}$$

The complexity is at least quadratic in the complexity of G_0. Therefore, it is rated as "failed".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a keyed pseudo-random function, H_K, which is not explicitly provided. Furthermore, it uses the assumption that EC encryption is secure.

Simplicity. This algorithm is about as complicated than the preceding algorithm, which was rated as MS.

confidentiality			yes: EC cryptosystem	good
completeness			yes: EC is public-key	good
indisting.	asymmetry		yes: only x is secret	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{7}{6}k}$	poor
		$\mathcal{C}_{G_1}$	> 0, bad distribution of n	poor
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, given π	poor
	GKR		yes	good
	side-channels	complexity	$t_e(G_1) = t_e$ $t_n(G_1) \approx t_n^2 \cdot T(H_K) + t_n \cdot T(\pi)$	failed
		memory	no	good
computational assumptions			EC + pseudo-random H_K, π	poor
simplicity			HS	failed

Table 6–9: Properties of Algorithm EC-SETUP of Figure 6–12. Properties are given with the trivial modification of the random oracle H being replaced by a family of pseudo-random functions, H_K.

6.3 Crépeau-Slakmon algorithms

The following five RSA backdoored key generators are from Crépeau and Slakmon [CS03]. This section is more detailed because a large portion of Chapter 7 is built on these algorithms.

These algorithms use a notation that involves two different pairs of RSA exponents, (d, e) and (δ, ϵ), per parameter n. As usual, suppose that $n = pq$ is an RSA modulus and denote a public key as (n, e) and the corresponding private key as d. Moreover, let (n, ϵ) be similar to a public key with the corresponding private key-like δ. Because the pair (n, ϵ) is not published, it is not a public key, but it is similar to one because it satisfies some properties of the proper public key (n, e), amongst other properties which are the main topic that this section develops. In general, (n, ϵ) is related to (n, e) in such a way that the designer can retrieve the concealed exponents, (ϵ, δ), from the public parameters, (n, e).

The algorithms from the paper of Crépeau and Slakmon rely on the following result of Wiener [Wie90] or some extension of this result. In Wiener's Theorem, the parameter $\kappa = (\epsilon\delta - 1)/\phi(n)$, where ϕ is the Euler totient function.

Theorem 6.3.1 (Wiener's low decryption exponent attack) *Any (n, ϵ) with $3\delta < n^{1/4}$ efficiently yields the values of κ and δ.*

Proof sketch: In Stinson [Sti06, Section 5.7.3], the proof is broken into three components. First, even if one removes the assumption that $3\delta < n^{1/4}$, it holds that,

$$\left|\frac{\epsilon}{n} - \frac{\kappa}{\delta}\right| < \frac{3\kappa}{\delta n^{1/2}}. \tag{6.5}$$

This is from applying the bounds on p and q to $(n - \phi(n))$ and recalling that it is assumed that $q < p < 2q$. Then the result is used in

$$\left|\frac{\epsilon}{n} - \frac{\kappa}{\delta}\right| = \left|\frac{1 + \kappa(\phi(n) - n)}{\delta n}\right|.$$

Note that $\kappa < \delta$, using $\epsilon \leq \phi(n)$ in $\kappa\phi(n) = \epsilon\delta - 1$. One takes

$$\kappa\epsilon \quad \leq \quad \kappa\phi(n) = \epsilon\delta - 1 < \epsilon\delta \tag{6.6}$$

and takes the LHS and RHS together:

$$\kappa\epsilon \quad < \quad \epsilon\delta \tag{6.7}$$

to simplify them into $\kappa < \delta$.

Second, given that $3\kappa < 3\delta < n^{1/4}$, then,

$$\left|\frac{\epsilon}{n} - \frac{\kappa}{\delta}\right| < \frac{1}{\delta n^{1/4}} \qquad \text{or} \qquad \left|\frac{\epsilon}{n} - \frac{\kappa}{\delta}\right| < \frac{1}{3\delta^2}. \tag{6.8}$$

Third, recall the well-known theorem that can be found in Hardy and Wright [HW79, Theorem 184] on the continued fraction expansion of x:

$$\text{“If } \left|\tfrac{\kappa}{\delta} - x\right| < \tfrac{1}{2\delta^2}\text{, then } \kappa/\delta \text{ is a convergent.”} \tag{6.9}$$

Given that $|\frac{\epsilon}{n} - \frac{\kappa}{\delta}| < \frac{1}{3\delta^2}$, one can efficiently find κ and δ (recall $\gcd(\kappa, \delta) = 1$). ■

Remark 6.3.2 *Note that $3\delta < n^{1/4}$ could be improved to $\sqrt{3/2}\,\delta < n^{1/4}$. However, it would improve our results about the number of generable keys, which are asymptotic and exponential, but by multiplicative constants.*

The usefulness of Wiener's Theorem and its variants is shown in Figure 6–13. Suppose that (δ, ϵ) is an easy to break instance of RSA keys and let (G_i, A_i) denote the pair of algorithms formed by the backdoored key generator and the corresponding retrieval algorithm, written by the designer. A general strategy is for G_i to transform this easy instance into another instance (d, e), so that (e, n) is published as the public key corresponding to d.

$$\begin{tikzcd}
(\delta,\epsilon) \arrow[r, "G_i"] & (d,e) \arrow[r, "\text{publication}"] & (e,n) \arrow[d, dotted, bend left, "A_i\ \text{(designer only)}"] \\
& & (\epsilon,n) \arrow[ull, dotted, "A_i\ \text{(easy instance)}"]
\end{tikzcd}$$

Figure 6–13: RSA backdoor generation from weak private key δ. The backdoored key generation process is represented by the full arrows. The key retrieval process comprises the entire cycle, except for the *publication* arrow.

This transformation is such that only the designer may find (ϵ, n) from (e, n), via the retrieval algorithm, A_i. Since (δ, ϵ) is an easy to break instance, it is trivial to find δ from (ϵ, n), and this is the second part of A_i. Its third and final part is, from (δ, ϵ) and then (p, q), to regenerate d in the same way as it done by G_i.

Therefore, the convention in notation is that the key generation algorithm returns (p, q, d, e) to the distinguisher while making sure to being able to trace back a related weaker tuple (p, q, δ, ϵ). The latter is weak in the sense that knowing its public part (n, ϵ) allows the designer to efficiently factorize $n = pq$ or, equivalently, to find d given e. Hence, the use of Wiener's Theorem and its variants is to ensure the completeness property of the Crépeau-Slakmon algorithms.

6.3.1 First algorithm: via Wiener's low decryption exponent attack

Throughout the remaining part of this chapter, $\pi_\beta : \mathbb{Z}_{\phi(n)} \rightarrow \mathbb{Z}_{\phi(n)}$ denotes an invertible one-way, pseudo-random permutation such that given β, computing π_β^{-1} is easy. The key to the permutation is the fixed secret key β, which remains the same for all legitimate users. This is simply coherent with symmetric cryptography, although it was not the case for the Anderson-Kaliski backdoor (Subsection 6.1.1): the secret key A is picked randomly for each legitimate user (*Symmetry* paragraph).

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$ is an assumed cryptographic permutation.

Algorithm CS-1 [RSA **key gen**]

```
1: Pick rp $p, q$ of size $k$ and set $n = pq$.
2: repeat
3:     Pick a random $\delta \in \mathbb{Z}^*_{\phi(n)}$ s.t. $\delta < n^{1/4}/3$.
4:     Compute $\epsilon \equiv \delta^{-1} \bmod \phi(n)$; $e = \pi_\beta(\epsilon)$.
5: until $\gcd(e, \phi(n)) = 1$.
6: Compute $d \equiv e^{-1} \bmod \phi(n)$.
7: return $(p, q, d, e)$.
```

Algorithm A-CS-1 [Key retrieval]

```
1: Input of $(n, e)$.
2: Factor $n = pq$ from $(n, \pi_\beta^{-1}(e))$ [Theorem 6.3.1].
3: Compute $d \equiv e^{-1} \bmod \phi(n)$.
4: return $d$.
```

Figure 6–14: [CS03]: a backdoor for RSA using weak Wiener keys and a permutation. The information on δ is embedded in e.

Algorithm properties. The algorithm, in Figure 6–14, illustrates the most basic application of the strategy of Figure 6–13.

Confidentiality. This property rests on the one-wayness of the permutation, π_β.

Completeness. This property is based on the classical Wiener's low decryption exponent attack (Theorem 6.3.1).

Symmetry. Algorithm (CS-1, A-CS-1) is a symmetric backdoor with secret key β.

Cardinality. The number of keys, per fixed n, is

$$\mathcal{N}_{G_1,e} = \#\{e\} = \#\{\epsilon\} = \#\{\delta\} = 2^{\frac{1}{2}k}.$$

In the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that $\mathcal{R}_{G_1} \approx 2^{-\frac{3}{2}k}$.

Distribution properties. The distribution of e is labeled "good", supposing that π_β is pseudo-random.

Entropy of embedding. Algorithm CS-1 is the first backdoored key generator, in its originally published form, to satisfy Definition 5.1.9 in the following way. It achieves $\mathcal{H}_{G_1} = 1$ with only explicit computational assumptions (on RSA), i.e. without computational assumptions on functions which are not explicitly provided. As illustrated in Figure 6–15, the backdoor information, δ, is uniformly spread throughout the n bits of e, given Assumption 2.2.22 (cf. Subsection 2.2.8).

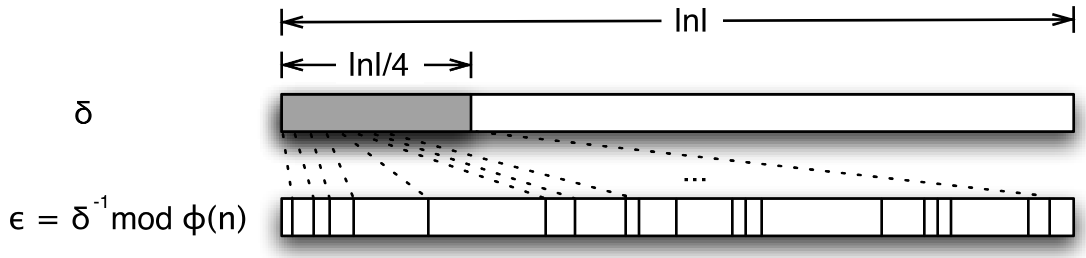


Figure 6–15: Entropy of CS-1 gained from the distribution properties of the RSA permutation. On the first line, the set of δ is generated from $k/2$ bits. On the second line, these bits are uniformly spread into $2k$ bits via the well pre-shuffled property of the RSA private exponent, given Assumption 2.2.17′.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. Step 3 and Steps 2 to 5 are two embedded loops that check the condition $\gcd(x, \phi(n)) = 1$, for a given x. Step 4 is an inversion modulo $\phi(n)$ which is included in the outside loop.

If Steps 3 and 4 run in time bounded above by

$$2t_e + T(\pi_\beta)$$

then Steps 2 to 5, in time bounded above by

$$t_e(2t_e + T(\pi_\beta)).$$

Overall:

$$\begin{aligned} T_n(G_1) &= t_n \\ T_e(G_1) &\approx t_e^2 + t_e \cdot T(\pi_\beta). \end{aligned}$$

The complexity is at least quadratic in the complexity of G_0. Therefore, it is rated as "failed".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

The results in *Entropy* depend on Assumption 2.2.22.

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			yes: one-way π_β	good
completeness			yes: Theorem 6.3.1	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{3}{2}k}$	**failed**
		$\mathcal{C}_{G_1}$	0, pseudo-random π_β	average
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	1, Assumption 2.2.22	good
	GKR		yes	good
	side channels	complexity	$T_n(G_1) = t_n$ $T_e(G_1) \approx t_e^2 + t_e \cdot T(\pi_\beta)$	**failed**
		memory	no	good
computational assumptions			one-way, pseudo-random π_β, Assumption 2.2.22	average
simplicity			LS	good

Table 6–10: Properties of Algorithm CS-1 of Figure 6–14.

6.3.2 Second algorithm: via upper bits of δ and prime ϵ

The second algorithm's completeness rests on the following theorem on partial information on the decryption exponent by Boneh, Durfree and Frankel (BDF).

Theorem 6.3.3 ([BDF98], Theorem 1.2, part 1) *Let $t \in [k/2, ..., k]$ and a prime $\delta \in [2^t, ..., 2^{t+1}]$. Given (n, δ) and $\epsilon\rceil^t$, one can factor n efficiently.*

In [CS03]'s usage, the partial information is on the hidden, weak, encryption exponent, ϵ. So the roles of δ and ϵ are swapped, w.r.t. the original Boneh, Durfee and Frankel (BDF) theorem.

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$ is an assumed cryptographic permutation.

Algorithm CS-2 [RSA key gen]

```
1: Pick rp $p, q$ of size $= k$ and set $n = pq$.
2: Pick a rp $\epsilon \in \mathbb{Z}^*_{\phi(n)}$ s.t. $|\epsilon| = k/2$.
3: Set $\delta \equiv \epsilon^{-1} \bmod \phi(n)$.
4: $\delta_H = \delta\rceil^{k/2}$.
5: repeat
6:     Pick random $r \in \{0,1\}^k$.
7:     $e = \pi_\beta(\delta_H : \epsilon : r)$.
8: until $\gcd(e, \phi(n)) = 1$.
9: Compute $d \equiv e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

Algorithm A-CS-2 [Key retrieval]

```
1: Input of $(n, e)$.
2: Compute $(\delta_H : \epsilon : r) = \pi_\beta^{-1}(e)$.
3: Compute $\delta$ from $(n, \delta_H, \epsilon)$ [Theorem 6.3.3].
4: Given $(\epsilon, \delta)$ factor $n$ as $p$,$q$.
5: Compute $d \equiv e^{-1} \bmod \phi(n)$.
6: return $d$.
```

Figure 6–16: [CS03]: a backdoor for RSA using a theorem of Boneh, Durfee, and Frankel and a permutation. The information on δ is embedded in e. Step 7 includes our modification discussed in *Cardinality*.

Algorithm properties. An unusual restriction is made on ϵ (which corresponds to δ, in the statement of Theorem 6.3.3) so that it is prime. This affects the cardinality of the backdoored key set, but not its indistinguishability. Indeed, the value corresponding to ϵ in the generation algorithm is encrypted via the pseudo-random π_β. The distinguisher cannot tell that it is unusually generated, unless π_β is broken.

Confidentiality. This property rests on the one-wayness of the permutation, π_β.

Completeness. This property is based on a theorem on partial information on the decryption exponent of BDF, Theorem 6.3.3.

Symmetry. Algorithm (CS-2, A-CS-2) is a symmetric backdoor with secret key β.

Cardinality. We directly analyze the algorithm with the trivial modification that we introduced. In Step 7 of G_1, we added the concatenation of k random bits. The original algorithm produces smaller e instead, and accordingly uses a π_β with smaller domain and image.

The number of keys, per fixed n, is $\mathcal{N}_{G_1,e} = \#\{e\} = \#\{\epsilon\} \cdot \#\{r\}$. By a modification of Example 2.2.7, for $k/2$-bit primes instead of k-bit, $\#\{\epsilon\} \approx 2^{k/2-\lg(k/2)}$. From Step 6 of CS-2, $\#\{r\} = 2^k$. In the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that

$$\mathcal{R}_{G_1} \approx \frac{2^{k/2-\lg k/2+k}}{2^{2k}} \approx 2^{-\frac{1}{2}k-\lg k}.$$

Distribution properties. This property and its argument are the same as for the preceding algorithm.

Entropy of embedding. This property and its argument are the same as for the preceding algorithm.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. The computation of n is the same as in G_0.

$$T_n(G_1) \;=\; t_n$$

The computation of e is similar to the one for the preceding algorithm, with three more operations. The first one is the generation of a $k/2$-bit random prime at Step 2, an operation which has complexity $t_{\sqrt{n}}$. Then Steps 5 to 7 form a loop that includes a gcd check and a cryptographic function π_β. Overall:

$$T_e(G_1) \;=\; t_{\sqrt{n}} + t_e + t_e \cdot T(\pi_\beta) \approx t_{\sqrt{n}} + t_e \cdot T(\pi_\beta).$$

These complexities are linear in the complexity of G_0 and $t_{\sqrt{n}}$ has significantly smaller complexity than RSA (the one of G_0) as a whole, of which t_e is a significant term. Assuming that $F = \pi_\beta$ is negligible with respect to t_e, Algorithm G_1 is rated "good". Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_1 is rated "good / F-relative".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

The results in *Entropy* depend on Assumption 2.2.22.

Simplicity. This algorithm is more complicated than the previous one, which was rated as LS. Compared to PAP, which was rated as LS-MS, it appears to have about the same level of structure.

confidentiality			yes: one-way π_β	good
completeness			yes: Theorem 6.3.3	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{1}{2}k-\lg k}$	poor
		$\mathcal{C}_{G_1}$	0, pseudo-random π_β	average
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	1, given Assumption 2.2.22	good
	GKR		yes	good
	side channels	complexity	$T_n(G_1) = t_n$ $T_e(G_1) \approx t_{\sqrt{n}} + t_e \cdot T(\pi_\beta)$	good / F-rel.
		memory	no	good
computational assumptions			one-way, pseudo-random π_β, Assumption 2.2.22	average
simplicity			LS-MS	average

Table 6–11: Properties of Algorithm CS-2 of Figure 6–16. Properties are given with the trivial modification explained in *Cardinality.*

6.3.3 Third algorithm: via upper and lower bits of δ

The algorithm's completeness rests on another theorem on partial information on the decryption exponent. Again, the notation is a little tricky. As for Theorem 6.3.3, in Crépeau and Slakmon's usage, the partial information is on the concealed, weak, encryption exponent, ϵ. The roles of δ and ϵ are swapped, w.r.t. the original theorem.

Theorem 6.3.4 ([BDF98], Theorem 4.6) *Let $t \in [1, ..., k]$ and $\delta \in [2^t, ..., 2^{t+1}]$. Given (n, δ), $\epsilon\rceil^t$ and $\epsilon\rfloor_{k/2}$, one can factor n efficiently.*

Theorem 6.3.4 compares to Theorem 6.3.3 as follows. In the notation of Crépeau and Slakmon, the requirement from the second algorithm, CS-2, that e be a prime is traded for the transmission of additional bits of information. As per Theorem 6.3.4, a message consisting of the $k/2$ lower bits of δ is sufficient.

PARAMETERS:
- β is a fixed secret key.
- $t \in [1, ..., k]$ is a fixed, non-secret, parameter.
- $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$ is an assumed cryptographic permutation.

Algorithm CS-3 [RSA key gen]

```
1: Pick rp $p, q$ of size $= k$ and set $n = pq$.
2: Pick a random $\epsilon \in \mathbb{Z}^*_{\phi(n)}$ s.t. $|\epsilon| = t$.
3: Set $\delta \equiv \epsilon^{-1} \bmod \phi(n)$.
4: $\delta_H = \delta\rceil^t$; $\delta_L = \delta\rfloor_{\frac{k}{2}}$.
5: repeat
6:     Pick a random $r \in \{0,1\}^{3k/2-2t}$.
7:     $e = \pi_\beta(\delta_H : \delta_L : \epsilon : r)$.
8: until gcd$(e, \phi(n)) = 1$.
9: Compute $d \equiv e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

Algorithm A-CS-3 [Key retrieval]

```
1: Input of $(n, e)$.
2: Compute $(\delta_H : \delta_L : \epsilon : r) = \pi_\beta^{-1}(e)$.
3: Compute $\delta$ from $(n, \delta_H, \delta_L, \epsilon)$ [Theorem 6.3.4].
4: Given $(\epsilon, \delta)$ factor $n$ as $p$,$q$.
5: Compute $d \equiv e^{-1} \bmod \phi(n)$.
6: return $d$.
```

Figure 6–17: [CS03]: a backdoor for RSA using another theorem of Boneh, Durfee, and Frankel and a permutation. The information on δ is embedded in e. Step 7 includes our modification discussed in *Cardinality*.

Algorithm properties. The same modification as in the preceding algorithm is applied.

Confidentiality. This property rests on the one-wayness of the permutation, π_β.

Completeness. This property is based on a theorem on partial information on the decryption exponent of BDF, Theorem 6.3.4.

Symmetry. Algorithm (CS-3, A-CS-3) is a symmetric backdoor with secret key β.

Cardinality. We directly analyze the algorithm with the trivial modification that we introduced. In Step 7 of G_1, we added the concatenation of $3k/2-2t$ random bits. The original algorithm produces smaller e instead, and accordingly uses a π_β with smaller domain and image.

The number of keys, per fixed n, is $\mathcal{N}_{G_1,e} = \#\{e\} = \#\{\epsilon\} \cdot \#\{r\}$. From Step 2 of CS-2, $\#\{\epsilon\} = 2^t$. From Step 6, $\#\{r\} = 2^{3k/2-2t}$. In the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that

$$\mathcal{R}_{G_1} \approx \frac{2^{t+3k/2-2t}}{2^{2k}} = 2^{-k/2-t}.$$

Distribution properties. This property and its argument are the same as for the preceding two algorithms.

Entropy of embedding. This property and its argument are the same as for the preceding two algorithms.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. The computation of n is the same as in G_0.

$$T_n(G_1) \;=\; t_n$$

The computation of e is similar to the one for the two preceding algorithms (without the additional generation of a random prime in the preceding algorithm). Overall:

$$T_e(G_1) \;\approx\; t_e + t_e \cdot T(\pi_\beta).$$

These complexities are linear in the complexity of G_0. Assuming that $F = \pi_\beta$ is negligible with respect to t_e, Algorithm G_1 is rated "good". Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_1 is rated "good / F-relative".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

The results in *Entropy* depend on Assumption 2.2.22.

Simplicity. This algorithm is about as complicated as the previous one.

confidentiality			yes: one-way π_β	good
completeness			yes: Theorem 6.3.4	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{1}{2}k-t}$	poor
		$\mathcal{C}_{G_1}$	0, pseudo-random π_β	average
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	1, given Assumption 2.2.22	good
	GKR		yes	good
	side channels	complexity	$T_n(G_1) = t_n$ $T_e(G_1) \approx t_e + t_e \cdot T(\pi_\beta)$	good / F-rel.
		memory	no	good
computational assumptions			one-way, pseudo-random π_β, Assumption 2.2.22	average
simplicity			LS-MS	average

Table 6–12: Properties of Algorithm CS-3 of Figure 6–17. Properties are given with the trivial modification explained in *Cardinality.*

Discussion. In the context of Crépeau and Slakmon's paper, the additional information to be embedded is not a disadvantage. This contrasts with the original interpretation of Theorem 6.3.4 in the paper of Boneh, Durfree and Frankel, where the non-consecutiveness of the bits of information was conceived to be a undesirable feature of such a theorem. Formally, CS-3 generates e as, for $t \in [1, ..., k]$:

$$e = \pi_\beta(\delta\rceil^t : \delta\rfloor_{k/2} : \underbrace{\epsilon}_{t \text{ bits}} : r), \tag{6.10}$$

with $r \in_U \{0,1\}^{3k/2-2t}$ and $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$.

Back to the general case, suppose that $t \in [1, ..., 3k/4]$ in Theorem 6.3.4. The size of the vector of variables to be transmitted is $\left|\delta\rceil^t : \delta\rfloor_{k/2} : \epsilon\right| = k/2 + 2t$ bits. Therefore a different algorithm could use a $\pi_\beta : \{0,1\}^{k/2+2t} \longrightarrow \{0,1\}^{k/2+2t}$. Denote by s the leftover space to append a prefix and a suffix of random bits to π_β:

$$s \;=\; 2k - \left(\frac{k}{2} + 2t\right) = 2\left(\frac{3k}{4} - t\right)$$

This different algorithm would generate e as:

$$e \;=\; r_1 : \pi_\beta(\delta_H : \delta_L : \epsilon) : r_2, \tag{6.11}$$

for a random $r_1 : r_2 \in \{0,1\}^{s/2}$. Also, $\pi_\beta : \{0,1\}^{2k-s} \to \{0,1\}^{2k-s}$.

Using Equation 6.11 instead of Equation 6.10 has immediate advantages. It allows the tuning of the distribution of e, if appropriate (independently from the choice of π_β). However, honestly generated e are distributed pseudo-randomly (as defined in theory), so there is no gain in indistinguishability. There may be an advantage in using Equation 6.11 in complexity. Suppose that π_β is more efficient on inputs of a certain length. Then Equation 6.11 is preferable, because it allows the designer to adjust input lengths.

Using Equation 6.11 also increases the cardinality because when t decreases by one, s increases by two. This gain is maximized when $t = 1$. Is taking $t = 1$ an improvement? Doing this reduces the number of possible pairs (δ, ϵ) to two. Moreover, the completeness of the backdoor rests on using $\delta\rfloor_{k/2}$, which is the same number of bits that the classical Wiener's low decryption exponent attack (Theorem 6.3.1) effectively embeds.

This situation of $t = 1$ makes CS-3 appear needlessly complicated, because it has more structure than CS-1. Moreover, the completeness of CS-3 is for the most part guaranteed by π_β being invertible, rather than via Theorem 6.3.4, since the latter is used on only two possible values of (δ, ϵ). Overall, this version encrypts nearly fixed information that allows to eventually factor n [11] . Nevertheless, this slightly different, if complicated and inelegant, version, that we now denote by G_1', does feature an interesting cardinality. Because $r_1 : r_2 \in \{0,1\}^{s/2}$ and $s = 2\left(\frac{3k}{4} - t\right) \leq 2\left(\frac{3k}{4} - 1\right)$, there are

$$\mathcal{N}_{G_1',e} \approx 2^{3k/2}$$

backdoored keys. This is the largest cardinality so far.

[11] Note that the required (approximate) $k/2$ bits is the same number as the one of bits of p that are required to factor n, via Coppersmith's Theorem 6.1.8. This is no coincidence since Theorem 6.3.4 rests on Theorem 6.1.8. The latter theorem is also used by one of Howgrave-Graham's backdoors (Figure 6–2).

6.3.4 Fourth algorithm: via hiding p in n

The fourth algorithm improves on both versions of PAP (Figures 6–5 and 6–10), in terms of indistinguishability and simplicity.

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \to \{0,1\}^{2k}$ is an assumed cryptographic permutation.

Algorithm CS-4 [RSA key gen]

```
 1: Pick a random $e \in \mathbb{Z}_{2^{2k}}$.
 2: Pick rp $p$ of size $= k$ s.t. $\gcd(e, p-1) = 1$.
 3: Pick a random odd $q'$ of size $= k$ set $n' := pq'$.
 4: Set $\tau = n'\rceil^{\frac{k}{4}}$ and $\mu = \pi_\beta(p\rceil^{\frac{k}{2}})$.
 5: while $\gcd(e, q-1) > 1$ or $q$ is composite do
 6:     Pick a random odd $\lambda \in \{0,1\}^{\frac{5k}{4}}$.
 7:     Set $n = (\tau : \mu : \lambda)$.
 8:     Set $q = \lfloor n/p \rfloor$.
 9: Compute $d = e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

Algorithm A-CS-4 [Key retrieval]

```
1: Input of $(n, e)$.
2: Compute $p\rceil^{\frac{k}{2}} = \pi_\beta^{-1}(n\rceil^{\frac{3k}{4}}\rfloor_{\frac{k}{2}})$.
3: Factor $n = pq$ [Theorem 6.1.8].
4: return $d$.
```

Figure 6–18: [CS03]: a backdoor for RSA using the hiding of p in n.

Algorithm properties. The key generation algorithm (Figure 6–18) produces instances of n that are the product of a truly random p and a somewhat random q such that:

- the upper $k/4$ bits of n have the correct distribution of such a product;
- the middle $k/2$ bits of n are an encryption of p's $k/2$ most significant bits;
- the lower $k/4$ bits of q are randomly chosen so that q is prime.

This configuration insures that approximately well distributed n and q can be found such that p is preserved in n. The length of the middle bits is determined by the minimum (known) number of bits of p that are sufficient to recover itself entirely (via Theorem 6.1.8). In the loop from Steps 5 to 8, the $k/4$ lower bits of q are effectively randomized until q satisfies some conditions. This randomization affects up to the (number of bits of p)$+k/4 = 5k/4^{\text{th}}$ bit of n (this is why the random choice of Step 6 is on these bits). Therefore, there remains $2k - 5k/4 = 3k/4$ bits to store information in n. The next $k/2$ bits are occupied by the information corresponding to p, as just described. After this, there remains $3k/4 - k/2 = k/4$ upper bits of n on which the only requirement is that they be distributed as a product of two prime numbers.

Confidentiality. This property rests on the one-wayness of the permutation, π_β.

Completeness. This property is based on Theorem 6.1.8.

Symmetry. Algorithm (CS-4, A-CS-4) is a symmetric backdoor with secret key β.

Cardinality. The number of n is counted. It is the concatenation of three parameters, $\tau{:}\mu{:}\lambda$, each being depend on p.

Count the number of μ, which is a permutation of $p\rceil^{\frac{k}{2}}$. Equivalently, count the number of $p\rceil^{\frac{k}{2}}$. Assuming that the bit distribution of primes p are uniform for a bit length of k, each k-bit string has a probability about

$$\frac{\pi(2^k)}{2^k} \approx 2^{-\lg k}$$

of being prime, by Example 2.2.6. We consider a total of $2^{k/2}$ such strings (and only their upper halves), so the expected number of those strings that are prime is about $2^{k/2-\lg k}$.

Count the number of τ. Not only μ depends on p, the two other parts of n do as well, since at the end, $p|n$. Thus, on Step 4, $\tau = n'\rceil^{\frac{k}{4}}$, of which one expects that $\frac{1}{2} \cdot \frac{k}{4} = k/8$ bits are free.

Count the number of λ. Before the loop from Steps 5 to 8, $\tau\!:\!\mu$ is fixed. At the end of the loop, a λ is selected such that $p|n$, i.e.

$$p \mid \tau\!:\!\mu\!:\!\lambda.$$

The ratio

$$n/p$$

has k bits. Because $p|n'$, the division in the $|\tau| = k/4$ upper bits carries through directly. The middle bits of n, denoted μ, are fixed, so only with negligible probability does the division within these bits carry through directly. Thus the choice of λ almost always compensates for the fixed μ. This means that the division by p affects

$$\begin{aligned} N &= |n/p| - |\tau| \\ &= k - \frac{1}{4}k \\ &= \frac{3}{4}k \end{aligned}$$

bits in λ. Therefore there remains

$$\begin{aligned} |\lambda| - N &= \frac{5}{4}k - \frac{3}{4}k \\ &= \frac{1}{2}k \end{aligned}$$

free bits in λ. The final number of λ is further reduced by the exit conditions of the loop. The first condition to satisfy, $\gcd(e, q-1) = 1$, does not significantly affect this number, as explained previously. The second condition that q is a prime accounts for the rejection of about $2^{\lg k}$ values, as previously argued. Therefore, the final number of λ is $2^{\frac{k}{2}-\lg k}$.

Summing up the numbers for the three parameters, $\tau : \mu : \lambda$, the generable number of n is

$$\begin{aligned} \#\{n\} &= \#\{\tau\} \cdot \#\{\mu\} \cdot \#\{\lambda\} \\ &\approx 2^{k/8} \cdot 2^{k/2-\lg k} \cdot 2^{k/2-\lg k} \\ &= 2^{\frac{9}{8}k-2\lg k}. \end{aligned}$$

In the ratio of the cardinalities of KSM and KS, the $\#\{e\}$ terms can be canceled so that

$$\mathcal{R}_{G_1} \approx 2^{-\frac{7}{8}k}.$$

Distribution properties. The key generation algorithm produces instances of n that are the product of a truly random p and a somewhat random q. Amongst other qualities, the upper $k/4$ bits of n have the correct distribution of such a product. The middle bits, μ, are distributed pseudo-randomly, given the assumption on π_β. Therefore, $\mathcal{C}_{G_1} \approx 0$, although further knowledge on the precise distribution of the product n should determine whether this value is truly negligibly small.

Entropy of embedding. At first, it seems that the entropy is bad: if p is fixed, μ is too. Improving the entropy requires only a minor modification to the generation algorithm. It suffices to use some of the bit length of the other parameters,

τ and λ, used to generate n within the generation of μ. Since the bits of τ are more significant, it is preferable to use the ones of λ, say $|r|$ bits. Then the middle part of n is generated as

$$\mu(p\rceil^{\frac{k}{2}} : r).$$

From the pseudo-randomness of the π_β function, the middle μ bits of n are pseudo-randomly distributed. The same does not apply to the lower bits λ, as they must be adjusted so that q is prime. The upper bits τ are by definition correctly distributed. This yields the evaluation: $0 < \mathcal{H}_{G_1} < 1$, given π_β.

Generalized key regeneration and variable correlation. This property is not satisfied unless some randomness is included in the generation of μ, as for the easily modified version of the PAP algorithm. This is the same modification that is suggested in *Entropy of embedding.*

Most cases are simple. If n is kept fixed then e is regenerated by simply being picked at random: $e \in_U \mathbb{Z}^*_{\phi(n)}$. If p (and possibly e) is kept, then it needs to be insured that the value of p is embedded in the new n. This is where the modification suggested in *Entropy of embedding* is useful. The bits r are regenerated and the value of

$$\mu(p\rceil^{\frac{k}{2}} : r)$$

changes so that coherent and new values of n and q can be generated from the same p as in the previous generation.

The only special case is when q (and possibly e) is kept, but p is regenerated. Then the roles of the primes are swapped, so that the old q is embedded in the new

n. The new p is generated in the same way as the old q had been, and becomes a function of this same old q.

Complexity. Suppose that the same modification that is suggested in *Entropy of embedding* is applied. First consider the generation of e. If it is regenerated alone, then it is simply picked at random: $e \in_U \mathbb{Z}^*_{\phi(n)}$. If it is generated with one or both of the other parameters, one or both of the gcd checks is done within the other parameter(s)' generation. In either case,

$$T_e(G_1) \;=\; t_e.$$

The most costly situation for the prime generation is when e is kept fixed: the complexity t_e is still accounted for because of the gcd checks of Steps 2 and 5. In fact, both steps combine one of these gcd checks with a primality check, which effectively multiplies their complexities. Overall, the worst case average complexities are:

$$T_p(G_1) \;=\; t_p t_e$$

and

$$\begin{aligned} T_q(G_1) &= t_{q'} + T(\pi_\beta) + t_q \cdot t_e \\ &= t_q \cdot t_e + T(\pi_\beta). \end{aligned}$$

The complexity is rated "poor" because it is more than linear but less than quadratic in the complexity of G_0.

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

Simplicity. This algorithm is simpler than PAP, which is rated LS-MS. However, its structure is not as close to the one of the honest algorithm as other algorithms.

confidentiality			yes: one-way π_β	good
completeness			yes: Corollary 6.1.8	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{7}{8}k-\lvert r\rvert}$	poor
		$\mathcal{C}_{G_1}$	0, or negl. small, pseudo-random π_β	good
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, pseudo-random π_β	poor
	GKR		yes	good
	side-channels	complexity	$T_p(G_1) = t_p t_e$ $T_q(G_1) = t_q \cdot t_e + T(\pi_\beta)$ $T_e(G_1) = t_e$	poor / F-rel.
		memory	no	good
computational assumptions			one-way, pseudo-random π_β	average
simplicity			LS	good

Table 6–13: Properties of Algorithm CS-4 of Figure 6–18. The analysis is done with the trivial modification suggested in *Entropy of embedding*.

6.3.5 Fifth algorithm: via Slakmon's variant of Wiener's Theorem

In [CS03, Section 5.1], n is instead used to encrypt the $k/3 \approx |n - \phi(n)| - 2k/3$ most significant bits of $n - \phi(n)$ (instead of $k/2$ msb's of p required by Coppersmith's method). Then a variant of Wiener's method, Theorem 6.3.5, allows to recover hidden exponents d up to size

$$|n - \phi(n)| - (2k/3)/2 \approx 2k/3.$$

This improves on both Wiener and Boneh-Durfee methods [BD00], because exponents d up to $n^{0.333}$ may be used and broken.

De Weger [dW02] and Slakmon [Sla00] have considered the algorithm of Wiener in a context where the primes p, q are partially known. In particular, [CS03] uses the following result of Slakmon.

Theorem 6.3.5 ([Sla00], Proposition 3.2.1) *Let t be an integer in the range $[\,1, ..., |n - \phi(n)|]$ and d be an integer in the range $[1, ..., 2^{|n-\phi(n)|-t/2}]$. Suppose we are given (n, e), and the $|n - \phi(n)| - t$ most significant bits of $n - \phi(n)$. Then we can factor n in time poly(k).*

Algorithm properties. Figure 6–19 is mostly the pasting together of Figures 6–18 (Algorithm CS-4) and 6–14 (Algorithm CS-1). However, as opposed to CS-4, both p, q' are picked as random primes. This is so that the most significant bits of

$$n' - \phi(n') = pq' - (p-1)(q'-1)$$

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$ is an assumed cryptographic permutation.

Algorithm CS-5 [RSA key gen]

```
1: Pick **rp** $p, q'$ of size $= k$ and set $n' := pq'$.
2: Set $\tau = n'\rceil^{\frac{2k}{3}}$ and $\mu = \pi_\beta\left((n' - \phi(n'))\rceil^{\frac{k}{3}}\right)$.
3: **while** $q$ is composite **do**
4:     Pick a random odd $\lambda \in \{0,1\}^k$.
5:     Set $n = (\tau : \mu : \lambda)$.
6:     Set $q = \lfloor n/p \rfloor$.
7: **repeat**
8:     Pick a random $\delta \in \mathbb{Z}^*_{\phi(n)}$ s.t. $\delta < n^{1/3}$.
9:     Compute $\epsilon \equiv \delta^{-1} \bmod \phi(n)$; $e = \pi_\beta(\epsilon)$.
10: **until** $\gcd(e, \phi(n)) = 1$.
11: Compute $d = e^{-1} \bmod \phi(n)$.
12: **return** $(p, q, d, e)$.
```

Algo. A-CS-5 [Key retrieval]

```
1: Input of $(n, e)$.
2: Compute $(n - \phi(n))\rceil^{\frac{k}{3}} = \pi_\beta^{-1}(n\rceil^k\rfloor_{\frac{k}{3}})$.
3: Factor $n = pq$ [Theorem 6.3.5].
4: **return** $d$.
```

Figure 6–19: [CS03]: a backdoor for RSA using the hiding of $n - \phi(n)$ in n.

approximate the ones of $n - \phi(n) = pq - (p-1)(q-1)$. Note that q' is prime is key for $\phi(n')$ to approximate n'. If q' is not prime, then $\phi(n')$ is not a factor of $q' - 1$.

By the choice of τ, it holds that $n\rceil^{2k/3} = n'\rceil^{2k/3}$, so $q\rceil^{k/3} \approx q'\rceil^{k/3}$ and

$$(n - \phi(n))\rceil^{2k/3} \approx (n' - \phi(n'))\rceil^{2k/3}.$$

Therefore, the equality holds for the latter with only $k/3$ bits. Consequently, in Step 4 of the generation algorithm, n's middle part is the encryption of $n - \phi(n)$.

Confidentiality. This property rests on the one-wayness of the permutation, π_β.

Completeness. This property is based on Theorem 6.3.5.

Symmetry. Algorithm (CS-5, A-CS-5) is a symmetric backdoor with secret key β.

Cardinality. Cardinality is accounted for similarly to the algorithm of Figures 6–18. The number of n is counted. It is the concatenation of three parameters, $\tau\!:\!\mu\!:\!\lambda$.

Count the number of μ, which is a permutation of $(n' - \phi(n'))\rceil^{\frac{k}{3}}$. Equivalently, count the number of $(n' - \phi(n'))\rceil^{\frac{k}{3}}$. Assuming that the bit distribution of primes p are uniform for a bit length of k, each k-bit string has a probability about

$$\frac{\pi(2^k)}{2^k} \approx 2^{-\lg k}$$

of being prime, by Example 2.2.6. The number $(n' - \phi(n'))\rceil^{\frac{k}{3}}$ is a function of the upper bits of the product of two such primes, so it is determined by each of their $k/6$ upper bits. This yields a total of $2^{\frac{k}{3}}$ such strings if primality was not involved, so the expected number of those strings that are prime is about $2^{\frac{k}{3}-\lg k}$.

Count the number of τ. Not only μ depends on p, the two other parts of n do as well, since at the end, $p|n$. Thus, on Step 4, $\tau = n'\rceil^{\frac{2}{3}k}$, of which one expects that $\frac{1}{2} \cdot \frac{2}{3}k = \frac{1}{3}k$ bits are free.

Count the number of λ. Before the loop from Steps 3 to 6, $\tau\!:\!\mu$ is fixed. At the end of the loop, a λ is selected such that $p|n$, i.e.

$$p \mid \tau\!:\!\mu\!:\!\lambda.$$

The ratio

$$n/p$$

has k bits. Because $p|n'$, the division in the $|\tau| = \frac{2}{3}k$ upper bits carries through directly. The middle bits of n, denoted μ, are fixed, so only with negligible probability does the division within these bits carry through directly. Thus the choice of λ almost always compensates for the fixed μ. This means that the division by p affects

$$\begin{aligned} N &= |n/p| - |\tau| \\ &= k - \frac{2}{3}k \\ &= \frac{1}{3}k \end{aligned}$$

bits in λ. Therefore there remains

$$\begin{aligned} |\lambda| - N &= k - \frac{1}{3}k \\ &= \frac{2}{3}k \end{aligned}$$

free bits in λ. The final number of λ is further reduced by the exit conditions of the loop. The first condition to satisfy, $\gcd(e, q-1) = 1$, does not significantly affect this number, as explained previously. The second condition that q is a prime accounts for the rejection of about $2^{\lg k}$ values, as previously argued. Therefore, the final number of λ is $2^{\frac{2}{3}k - \lg k}$.

Summing up the numbers for the three parameters, $\tau : \mu : \lambda$, the generable number of n is

$$\begin{aligned} \#\{n\} &= \#\{\tau\} \cdot \#\{\mu\} \cdot \#\{\lambda\} \\ &\approx 2^{\frac{1}{3}k} \cdot 2^{\frac{k}{3} - \lg k} \cdot 2^{\frac{2}{3}k - \lg k} \\ &= 2^{\frac{4}{3}k - 2\lg k}. \end{aligned}$$

Finally, because of the combination of the two methods, the number of e is also accounted. It holds that $\#\{e\} = \#\{\delta\} = 2^{2k/3}$, similarly as for the first method for which this number is $2^{k/2}$.

In the ratio of the cardinalities of KSM and KS, no pair of factors can be exactly canceled. Overall,

$$\begin{aligned} \mathcal{R}_{G_1} &\approx 2^{-\frac{2}{3}k} \cdot 2^{-\frac{4}{3}k} \\ &= 2^{-2k}. \end{aligned}$$

Comparison. Compare the cardinality of this algorithm, CS-5, with the preceding one, CS-4. CS-5 has a higher approximate $\#\{n\}$, because $2^{\frac{4}{3}k} > 2^{\frac{9}{8}k}$. This is because CS-5 reduces the amount of backdoor information to be encrypted in n, even though this information also fixes other parameters in the cryptosystem, p for instance. However, CS-5 has a lesser $\#\{e\}$, since d is constrained via the acceptable size of δ.

Distribution properties. This property and its argument are similar to the ones for the preceding algorithm. Nevertheless, CS-5 increases the security of the hidden prime method, because it leaves more bits that can be distributed appropriately in n.

Entropy of embedding. The same analysis as the preceding algorithm applies.

Generalized key regeneration and variable correlation. The analysis does not require the modification applied to the preceding algorithm, CS-4. As the result of the choice of backdoor information, CS-5 immediately satisfies GKR: if only one of the primes changes, μ changes as well.

Concretely, consider the case when p is fixed and a new q (and possibly e) is requested. The backdoor information to be embedded is $\mu = \pi_\beta\left((n' - \phi(n'))\rceil^{\frac{k}{3}}\right)$, where $n' = pq'$, so there are many values of μ associated to the same p. The generation algorithm (likely) directly produces new and independent μ and then q.

Even if δ is of a special form, other parameters that are usually independent remain such, and this has no effect on GKR.

Complexity. As the result of the combination of two algorithms, CS-5 suffers the high computational complexity of the first algorithm, CS-1. The complexity is similar to the one of the algorithm of Figure 6–14. For the generation of e:

$$\begin{aligned} T_e(G_1) &\approx t_e(t_e + T(\pi_\beta)) \\ &= t_e^2 + t_e \cdot T(\pi_\beta). \end{aligned}$$

For the generation of n:

$$T_p(G_1) = t_p$$

and

$$\begin{aligned} T_q(G_1) &= t_{q'} + T(\pi_\beta) + t_q \\ &= t_q + T(\pi_\beta). \end{aligned}$$

The complexity is rated "failed" because it is quadratic in the complexity of G_0.

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

Simplicity. As the result of the combination of two algorithms, that the backdoor relies on special forms of both n and e yields increased structure (less simplicity). In particular, π_β is used in both public parameters.

CS-5 is more complicated than CS-4, which is rated as LS-MS. This is due to steps in the generation of e, in addition to the steps in the generation of n, which are similar in both algorithms.

confidentiality			yes: one-way π_β	good
completeness			yes: Theorem 6.3.5	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_1}$	2^{-2k}	**failed**
		$\mathcal{C}_{G_1}$	0, or negl. small, pseudo-random π_β	good
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, pseudo-random π_β	poor
	GKR		yes	good
	side channels	complexity	$T_p(G_1) = t_p$ $T_q(G_1) = t_q + T(\pi_\beta)$ $T_e(G_1) \approx t_e^2 + t_e \cdot T(\pi_\beta)$	**failed**
		memory	no	good
computational assumptions			one-way, pseudo-random π_β	average
simplicity			MS	average

Table 6–14: Properties of Algorithm CS-5 of Figure 6–19.

Discussion on the optimal length of n's parameters. The goal is to determine the optimal parameter lengths, that is, the ones for which the cardinality of the backdoored key set is maximized. For this purpose, suppose that $n' = pq'$ and n are generated similarly as in the preceding two backdoor algorithms, but, for simplicity, the validity probability when sampling prime numbers is ignored.

Suppose that $n = \tau : \mu : \lambda$, with $|\tau| = vk$, $|\mu| = xk$, and $|\lambda| = wk$, for parameters v, x and $w \in \mathbf{R}$ such that $x, v, w \geq 0$ and $x + v + w = 2$. Let $\tau = n' \rceil^{vk}$. Let μ fix xk bits of n and suppose that, as in the previous two algorithms, fixing μ determines half the bits of τ. Let λ be tunable in order to insure that $n = pq$. Therefore, there remains $wk - (k - |\tau|) = (w + v - 1)k$ undetermined bits in λ.

The total number of free bits in each parameter is

$$\begin{aligned}
|\tau| &= \frac{v}{2}k \\
|\mu| &= xk \\
|\lambda| &= (w + v - 1)k.
\end{aligned}$$

Finally

$$\begin{aligned}
|n| &= \left[\frac{v}{2} + x + (w + v - 1)\right] k = \left(x + w + \frac{3}{2}v - 1\right) k \\
&= \frac{v}{2} + 1.
\end{aligned}$$

Therefore, for best distribution of n and the maximum cardinality, the designer should choose τ as large as possible. This is related to choosing μ as small as possible, because it leaves more space for the other parameters. However, $|\mu| = xk$ is dependent on available theorems, so let the minimal known x be fixed. For λ, the designer insures that w is such that $|\lambda| = (w + v - 1)k > 0$, where the inequality is strict so that many choices of LSB of n are produced. Suppose that $|\lambda| = \varepsilon k$. Then the optimal $v = 2 - x - \varepsilon$ and the corresponding number of generable n is

$$\#\{n\} = 2^{2-(x+\varepsilon)/2}.$$

6.3.6 Choices of permutation, π_β

Throughout our exhibition of Crépeau and Slakmon's results, the choice of π_β was left open. Nevertheless, suggestions are included in the original paper. Because some results of Chapter 7 are extensions of these suggestions, we exhibit and comment them in this section.

Hidden (δ, ϵ) algorithms. As usual, let $e = \pi_\beta(\epsilon)$. [CS03] suggests three types of π_β.

1. The first one consists in the XORing with the secret key, β, or a simple modification of it (to match lengths).

   $$\pi_\beta(x) = x \oplus (2\beta)\rfloor_{|x|}$$

   taking into account that the LSB needs not be flipped, so $\beta\rfloor_1 = 0$.
2. The second one consists in using cryptographic primitives.

   $$\pi_\beta(x) = \text{DES}_\beta(x) \text{ or } \pi_\beta(x) = \text{AES}_\beta(x)$$

   but these introduce more computational complexity and extra computational assumptions. To avoid this, one can use a more RSA-like permutation:

   $$\pi_\beta(x) \equiv x^{-1} \bmod \beta$$

   where β is prime or a product of a small number of primes such that π_β^{-1} can easily be computed.
3. The third suggestion consists in different variations of additions and modulo operations involving β.

(a) If the modulo changes with each n, then it is more indistinguishable. If an even modulo such as $n+1$ is taken, then the odd integers are mapped to themselves. Since $(n+1)-\phi(n)=p+q$ which is exponentially small w.r.t. $\phi(n)$, the probability that a permutation modulo $n+1$ maps an element to a value greater than $\phi(n)$ is negligible. For $\frac{1}{2}\max n \leq \beta \leq \max n$, let

$$\pi_\beta(x) \equiv (x+2\beta) \bmod (n+1).$$

However, this was broken by Vaudenay [Vau]. Suppose a small prime r such that

$$r|\phi(n) \text{ and } r|n+1$$

then this [12] implies $\gcd(\epsilon, r) = 1$ and $\gcd(e, r) = 1$. In particular, $\gcd(\epsilon, r) \neq 0$, so there exists a small value, $e < r$, such that the following holds:

$$e \not\equiv 2\beta \bmod r$$

unless $r|2\beta$, in which case

$$e \equiv \epsilon \bmod r$$

which is uniformly distributed, so unlikely to happen.

Unless this unlikely case holds, it suffices for the distinguisher to test a small number of $(e, \phi(n), n)$ in order to tell backdoored keys apart. Pick a

[12] For non-backdoored keys, this is true with probability about $1/r^2$. Then a distinguisher can expect to efficiently test what follows.

small prime r such that $r|\phi(n)$ and $r|n+1$. If the keys are non-backdoored, then with probability about $1/r^2$, it holds that $e \not\equiv 2\beta \bmod r$, so the values of e modulo r are uniformly distributed and in particular, do not avoid any given value. If the keys are backdoored, then it holds that $e \not\equiv 2\beta \bmod r$, so there exists a value of e modulo r which is constantly avoided.

(b) A more general permutation can use extra hidden parameters. Notice that $n+1-2\sqrt{n}$ is always an upper bound on $\phi(n)$ and thus

$$\pi_{\beta,\mu}(x) = (x+2\beta) \bmod (n+1-2m)$$

may also be used, where $m = \mu \bmod \lfloor\sqrt{n}\rfloor$ for a fixed μ, such that $\sqrt{\max n} \le \mu \le 2\sqrt{\max n}$, i.e., μ is an arbitrary constant at least half the size of the largest generable n.

(c) Adding secret parameters may however be the cause of distinguishability. Let α be such that $1 \le \alpha \le \max n$ and generalize $x+2\beta$ to

$$\pi_{n,\alpha,\beta,\mu}(x) = ((2\alpha+1)x+2\beta) \bmod (n+1-2m).$$

If $\gcd(2\alpha+1, \phi(n), n+1-2m) > 2$, given two different sets of exponents, but with the same modulus (p,q,e,d) and (p,q,e',d'), a distinguisher can compute $e'-e$ which is the same as $(2\alpha+1)(\epsilon'-\epsilon)$ up to a multiple of $(n+1-2m)$. The distinguisher could then notice that $\gcd(e'-e,\phi(n)) > 2$ all the time which is unusual, despite the fact that μ and β are unknown.

Summing up, suggestions 1, 2 and 3(b) are the most promising. Another example of potentially useful non-algebraically-based functions are bit permutations.

However, they turn out to be too complicated, if portions of the bits are constant, which may be the case in particular with GKR. A keyed permutation would be necessary, and this suffers complications similar to a pseudo-random function.

Hidden p algorithm. In this case, let $n = \pi_\beta(p)$. The preceding suggestion 1 cannot be used because XORing two instances of backdoored keys, (n, p) and (n', p'), allows to match strings as follows.

$$(n' \oplus n)\rceil^{\frac{3k}{4}}\rfloor_{\frac{k}{2}} = (p' \oplus p)\rceil^{\frac{k}{2}}$$

The second suggestion's RSA-like permutation cannot be used directly, because it yields multiples of β, hence after few trials, β itself via gcd computations. Mathematically, this is:

$$n\rceil^{\frac{3k}{4}}\rfloor_{\frac{k}{2}} p\rceil^{\frac{k}{2}} - 1 = (p\rceil^{\frac{k}{2}^{-1}} \bmod \beta) \cdot p\rceil^{\frac{k}{2}} - 1 = k\beta + 1 - 1$$

where $k \geq 0$ is a constant.

An effective counter-measure is to pad $p\rceil^{\frac{k}{2}}$ with random bits, as it was done to improve entropy, in the Discussion paragraph of Subsection 6.3.4. Mathematically:

$$n = \pi_\beta(p : r)$$

for some random bits r.

In [CS03], a modification of suggestion 3(b) is preferred. It consists in computing of a modular inverse modulo a fixed predetermined prime near $2^{\frac{k}{4}}$ and XORing with a fixed string:

$$\pi_{\beta,\mu}(x) = \left(x \oplus (2\mu)\rfloor_{|x|}\right)^{-1} \bmod \beta$$

or

$$\pi_{\beta,\mu}(x) = \left(x^{-1} \bmod \beta\right) \oplus (2\mu)\rfloor_{|\beta|}.$$

This thwarts both preceding problems, but suffers from a different problem. For given β, μ, there are a lesser number of choices of q than normally. The distinguisher may be able to tell that q's distribution is unusual. This happens with a lesser chance if the preceding suggestion, with padding, is used. Mathematically:

$$\pi_{\beta,\mu}(x) = \left(x \oplus (2\mu)\rfloor_{|x|}\right)^{-1} \bmod \beta : r$$

for some random bits r, or

$$\pi_{\beta,\mu}(x) = \left(x^{-1} \bmod \beta\right) \oplus (2\mu)\rfloor_{|\beta|} : r.$$

for some random bits r.

6.4 Chapter notes

Section 6.1. Simple algorithms. The notes are divided per subsection.

Subsection 6.1.1. Anderson-Kaliski backdoors. The masters project of J. Cho covers the Anderson backdoor in details [Cho04, Section 2]. A statement and proof of a special case of Theorem 6.1.5, also known as *Dirichlet's Theorem on Primes in Arithmetic Progressions*, is found in [DF91, p.469]. The well-known theorem on the continued fraction expansion, given as Theorem 6.1.1, can be found stated and proven as [HW79, Theorem 184].

Subsection 6.1.2. Howgrave-Graham backdoors. Corollary 6.1.9 is implicit in [HG01, p. 52]. Both backdoors of this section are very implicitly described (backdoors are not the main topic of this paper).

Section 6.3. Crépeau-Slakmon algorithms. In the original paper of Crépeau and Slakmon [CS03]:

- Algorithm (CS-1, A-CS-1) is denoted as (CS-3.1, CS-3.2);

- Algorithm (CS-2, A-CS-2) is denoted as (CS-3.2, CS-3.4);

- Algorithm (CS-3, A-CS-3) is denoted as (CS-3.5, CS-3.6);

- Algorithm (CS-4, A-CS-4) is denoted as (CS-5.1, CS-5.2);

- Algorithm (CS-5, A-CS-5) is described in Section 5.1 of that paper.

Subsection 6.3.5. Fifth algorithm: via Slakmon's variant of Wiener's Theorem. The backdoor of this section is implicitly described in [CS03], probably because it bears much resemblance to other explicitly described backdoors.

Subsection 6.3.6. Choices of permutation, π_β. Most of the information is taken from [CS03, Sections 4 and 5] as well as [Vau].

CHAPTER 7
Improved algorithms

In this chapter, we present new backdoored key generators that supersede the existing ones. Our generators improve on three critical aspects which were introduced in Chapter 5 and used in the previous chapter to compare the previous works.

Diversity and complexity. Consider the RSA backdoored key generators of Chapter 6 that passed all the criteria of Chapter 5 [1] . All of these algorithms have a quadratic time complexity and some have a low cardinality as well. We present the first generator that passes all the criteria **and** that features a linear time complexity. Comparable improvements for EG backdoored key generation are also shown.

Generalized key regeneration. The first EG backdoored key generators that support generalized key regeneration are presented [2] . Of course, these generators also pass all the other criteria.

Asymmetric encryption function. The first RSA and EG backdoored key generators that pass all the criteria **and** that use an asymmetric encryption function are presented.

[1] One could make abstraction of the entropy measure, $\mathcal{H}_{G_1}$, as it is more an additional desirable feature than a failure criterion. However, in the documented algorithms, failure of this criterion is always paired with the failure of other criteria.

[2] For RSA, this was achieved in [CS03].

7.1 First improvement: complexity, then diversity

In this section, we use a new extension of Wiener's Theorem and a generalization of a theorem of Boneh, Durfree and Frankel, itself based on Coppersmith's lattice factorization, to obtain an RSA backdoored key generator with better complexity and diversity. Firstly, this new generator produces keys in time linearly related to the time of the honest algorithm. Secondly, the generator can produce a greater number of generable keys that are at least as secure, thus increasing the key diversity. For a fixed "backdoor secret key" (the information held by the party making the backdoors), the diversity of backdoored public keys is improved in that they are closer to the distribution of honest public keys.

Our results improve directly on the results of Crépeau and Slakmon, in particular [CS03, Algorithms (3.1, 3.2) and (3.5, 3.6)]. These algorithms are given in Subsections 6.3.1 and 6.3.3, respectively, and denoted as (CS-1, A-CS-1) and (CS-3, A-CS-3). In this series of results, the publicly known e is typically a private function of an unknown, hidden ϵ. Learning ϵ typically implies the discovery of δ and d.

7.1.1 Outline of the first improvement

Subsections 7.1.3 and 7.1.4 present four new hidden exponent methods for an RSA key generation algorithm. The first algorithm, which is the simplest, serves to demonstrate the achievable computational complexity of this type of algorithms. The time complexity of all four algorithms is the same as that of the first one (Subsection 7.1.3), as they are very closely related.

First, we show the following proposition. It is an improvement on previous comparable backdoored key generators.

Proposition 7.1.9 [Running time - informal] *It is possible to generate cheating keys in time linear in the honest algorithm's running time.*

The second and third algorithm are meant to build intuition, leading to the fourth one. This is the strongest of the results, and uses no additional cryptographic assumptions. The first two pairs of algorithms, (G_1, R_1) and (G_2, R_2), yield simple, general backdoor methods with a choice of permutation function. The other two pairs of algorithms, (G_3, R_3) and (G_4, R_4), are consecutive generalizations of [CS03]. Alternatively, this result can be seen as an improvement on Wiener's attack on RSA small public exponents, and more precisely on the one of Boneh, Durfree and Frankel that uses parts of the private exponent [BDF98].

Proposition 7.1.10 [Diversity of backdoor keys I - informal] *Within this time constraint it is possible to generate about $n^{3/4}$ keys, and this is closer to the total number of possible honest keys than before. This refers to the number of public/private keys generable per secret key of a backdoored key generator (as it is usual, thus far).*

Proposition 7.1.14 [Diversity of backdoor keys II - informal] *This second result is improved, because for a different secret key β, the generated keys are almost distinct. Thus, almost all keys can be backdoored, so for a random β, the backdoored keys are distributed closely to the honest ones. This is not achieved for a fixed β, in which case Proposition 7.1.10 applies.*

As explained in details in Subsections 7.1.4 and 7.2.4, the resulting diversity is increased. This comprises the number and distribution of the generable keys. The last two propositions imply that the backdoored keys reach almost all honest keys.

$$\begin{tikzcd}
 & & \text{running time (Proposition 7.1.9)} \\
 & \text{direct results} \arrow[ur] \arrow[dr] & \\
\text{Wiener's Theorem} \arrow[ur] \arrow[dr] & & \text{key diversity (Proposition 7.1.10)} \\
 & \text{Theorem 7.1.3} \arrow[r] & \text{improved diversity (Prop. 7.1.14)}
\end{tikzcd}$$

Figure 7–1: Illustration of the first sequence of improvements. Compare with Figure 7–10.

Subsection 7.1.2 presents two new extensions of Wiener's Theorem for RSA small private exponents, on which all four algorithm that follow rely. The generation algorithm G_i, creates a random backdoored pair of private and public exponents, (δ, ϵ), with small δ, and transforms it into a random-looking private and public exponent pair, (d, e). This pair is related to (δ, ϵ) in a secret way that only the designer may invert with the retrieval algorithm A_i. From public knowledge of (e, n) only, the designer may recover ϵ, break the easy instance (ϵ, n), and compute δ. The factorization of n is obtained via standard techniques, once δ and ϵ are known: (expectedly) efficiently via a probabilistic algorithm [Mil75], and in polynomial time although quite inefficiently via a deterministic algorithm [May04]. This is the same as in the [CS03] algorithms, as detailed and illustrated in Subsection 6.3.1.

Theorems 7.1.3, 7.1.7 [Generalizations of Wiener's Theorem - informal] *Suppose that δ is small enough, and that n as well as an additive approximation ϵ*

to e are given. Then one may efficiently find κ (defined as $\kappa = \frac{\epsilon\delta - 1}{\phi(n)}$) as well as δ. Furthermore, if δ is prime, then one may efficiently factor n.

At first, the algorithms may seem very intricate, especially considering that there have been simpler proposals. Nevertheless, as our algorithms avoid leaving all the security (i.e., the indistinguishability issues) to an encryption function, they require less structure. By using "trivial" encryption functions, we obtain more direct results. This however does not rule out the existence of simpler algorithms. More properties of backdoored key generators are considered in order to differentiate them in terms of strength and quality in Section 7.5.3

Finally, our best result, given by algorithm (G_4, R_4) relies on less computational hypotheses. More precisely, Algorithm G_3 leaves the choice of permutation open, while G_4 incorporates a specific permutation with desirable properties.

7.1.2 Preliminary results: new extensions of Wiener's Theorem

Each result in this section uses Wiener's Theorem [Wie90], in one form or another. A sketch of proof of this well-known theorem is given at the beginning of Section 6.3. Recall that κ is such that

$$\epsilon\delta - \kappa\phi(n) = 1.$$

It is a parameter which is useful in Wiener's Theorem (Theorem 6.3.1), as well as in some of our propositions.

Following is a first generalization of Wiener's Theorem 6.3.1, from Blömer and May [BM04, Theorem 2]. It deals with cases where ϵ is such that

$$\epsilon\delta \equiv c \bmod \phi(n),$$

where c is generally not equal to 1. The significance of this generalization is that it works for all exponents

$$\epsilon \equiv c\delta^{-1} \bmod \phi(n),$$

where c and δ are sufficiently small. As shown in that paper, it follows that there are at least $n^{3/4}$ such "weak keys" and that this number increases with decreasing prime difference $p-q$.

Theorem 7.1.1 ([BM04], Generalized Wiener's Theorem) *Suppose that $\gamma \leq 1$ and $p-q \geq \gamma n^{1/2}$. Given an $\epsilon \in \mathbb{Z}^*_{\phi(n)}$, that satisfies*

$$\epsilon\delta \equiv c \bmod \phi(n)$$

such that

$$0 < \delta \leq \frac{1}{3}n^{1/4} \text{ and } |c| \leq \gamma n^{-3/4}\epsilon\delta$$

then n can be factored efficiently.

The conditions of Theorem 7.1.1 imply that $\epsilon\delta - c \neq 0$, therefore excluding trivial congruences: since $\gamma \leq 1$, then $|c| < \epsilon\delta$. Letting $\epsilon\delta - c = \kappa\phi(n)$, this in turn implies $\kappa > 0$.

The main components of this generalization's proof are reproduced here. They are useful for reference in the proof of Theorem 7.1.5.

Proof outline:

- The unknown parameters δ, κ can be found among the convergents of the continued fraction expansion of ϵ/n (e.g. [HW79, Theorem 184]).
- From δ and κ, an approximation of $p+q$ is computed.

- From an approximation of $p+q$, an approximation of $p-q$ is computed.
- Combining both approximations yields an approximation of p, which in turn leads to the factorization of n via Coppersmith's Theorem (Corollary 6.1.8). ■

In fact, one of the hypotheses of Theorem 7.1.1 is not required. The first component of the preceding outline is proven without using that $\epsilon \in \mathbb{Z}^*_{\phi(n)}$. This is useful for building backdoored key generators. This proof is reproduced following the updated theorem.

Theorem 7.1.2 (Generalized Wiener's Theorem, updated) *The statement of Theorem 7.1.1 holds for $\epsilon \in \mathbb{Z}_{\phi(n)}$ in general.*

Proof of the outline's first component: An $\epsilon \in \mathbb{Z}_{\phi(n)}$ that satisfies $\epsilon\delta \equiv c \bmod \phi(n)$ is given. Note that if $\gcd(\epsilon, \phi(n)) = x \neq 1$, then $x|(\epsilon\delta - \kappa\phi(n)) = c$. This does not affect the following.

Via the definition of $\phi(n)$, one has

$$\epsilon\delta - c = \kappa\phi(n) = \kappa(n - p - q + 1) \tag{7.1}$$

and, dividing the LHS and RHS by $n\delta$, one obtains

$$\frac{\epsilon}{n} - \frac{\kappa}{\delta} = -\frac{\kappa(p+q-1) - c}{n\delta}.$$

If δ and c are the minimum values that satisfy $\epsilon\delta \equiv c \bmod \phi(n)$ then $\frac{\kappa}{\delta}$ is a fraction in lowest term. To see this, assume that it is not the case, so $\gcd(\kappa, \delta) = x \neq 1$. Then Equation 7.1 is divisible by x and therefore δ and c are not minimal, which is a contradiction.

Then, by the continued fraction theorem (e.g. [HW79, Theorem 184]), the fraction $\frac{\kappa}{\delta}$ appears in the continued fraction expansion of $\frac{\epsilon}{n}$ if

$$\left|\frac{\epsilon}{n} - \frac{\kappa}{\delta}\right| < \frac{1}{2\delta^2}.$$

It remains to show that $|\kappa(p+q-1) - c| < \frac{n}{2\delta}$. We have

$$\begin{aligned}
\kappa(p+q-1)+c &\leq \frac{10}{4}\frac{\epsilon\delta}{\phi(n)}n^{1/2} + \gamma n^{-3/4}\epsilon\delta && (7.2)\\
&\leq \frac{10}{4}\delta n^{1/2} + \delta n^{1/4} \\
&\leq 3\delta n^{1/2} && (7.3)
\end{aligned}$$

where Inequality 7.2 comes from $p+q \leq 2n^{1/2}$, the bound on c in hypothesis, and the following approximation of κ. This variable is isolated as

$$\kappa = \frac{\epsilon\delta + c}{\phi(n)}$$

and approximated with $|c| \leq \gamma n^{-3/4}\epsilon\delta$ and $n \geq 2^{12}$, yielding

$$\kappa \leq \frac{5}{4}\frac{\epsilon\delta}{\phi(n)}.$$

Inequality 7.3 also holds for $n \geq 2^{12}$. That bound on n comes from that we can assume that $n \geq (8/c)^4$, otherwise $p - q \in \mathcal{O}(n^{1/4})$ in which case Fermat's factorization algorithm succeeds in polynomial time.

It remains to show that the RHS of Inequation 7.3 is bounded above by

$$3\delta n^{1/2} < \frac{n}{2\delta} \quad \text{i.e.} \quad \delta \leq \frac{1}{\sqrt{6}}n^{1/4}.$$

The bound on δ in hypothesis is slightly stronger: $\delta \leq \frac{1}{3}\, n^{1/4}$. So the requirement of the theorem on continued fractions (e.g. [HW79, Theorem 184]) is fulfilled. ■

A new extension to Wiener's Theorem is hereby contributed. This result is the cornerstone on which new backdoored key generators rely. Nevertheless, the result was deemed an *extension*, not a *generalization*, because it is easily deducible from Wiener's Theorem, even though it is comparable and complementary to Theorem 7.1.1. Suppose that the public exponent e is an approximation to a concealed exponent ϵ, so only e is known. In the case that e approximates ϵ sufficiently closely, one can efficiently derive (κ, δ) from (n, e).

In the following theorem, the parametrization with α, ℓ_1 and ℓ_2, is intended to make the theorem and the derived algorithms as general as possible. For simplicity, one can picture the following statement with $\ell_1 \approx 1/3$, $\ell_2 = 1$ and $\alpha = 1/4$.

Theorem 7.1.3 (An extension of Wiener's Theorem) *Let $\alpha \leq 1/4$, $\ell_1 < \frac{1}{3}$ and ℓ_2 s.t. $\ell_1^2(\ell_2 + 3) \leq \frac{1}{3}$. Any (n, e) related to (n, ϵ), with concealed ϵ, that satisfies*

$$|e - \epsilon| < \ell_2\, n^{1-2\alpha}$$

such that

$$\delta < \ell_1\, n^{\alpha}$$

efficiently yields the values of κ and δ.

Proof: From Equation 6.5 in the proof of Theorem 6.3.1, and since $\kappa < \delta$ and $\alpha \leq 1/4$, then

$$\left|\frac{\epsilon}{n} - \frac{\kappa}{\delta}\right| < \frac{3\kappa}{\delta n^{1/2}} < \frac{3\ell_1}{\delta n^{1/4}}$$

holds, so

$$\begin{aligned}
\left|\frac{e}{n}-\frac{\kappa}{\delta}\right| &\leq \frac{|e-\epsilon|}{n}+\left|\frac{\epsilon}{n}-\frac{\kappa}{\delta}\right| \\
&< \frac{\ell_2}{n^{2\alpha}}+\frac{3\ell_1}{\delta n^{1/4}} \\
&< \frac{\ell_1^2}{\delta^2}\left(\ell_2+3\right) \qquad (7.4) \\
&\leq \frac{1}{3\delta^2} \qquad (7.5)
\end{aligned}$$

where Inequality 7.4 holds because

$$\frac{1}{n^{1/4}} \leq \frac{1}{n^{\alpha}} < \frac{\ell_1}{\delta},$$

and where Inequality 7.5 holds by hypothesis. From Equation 6.9 in the proof of Theorem 6.3.1, this is sufficient to efficiently yield κ and δ. ■

Theorem 7.1.4 (An extension of Wiener's Theorem, updated) *The statement of Theorem 7.1.3 holds for $e \in \mathbb{Z}$ in general.*

Proof: That $e \in \mathbb{Z}^*_{\phi(n)}$ is not used in the previous proof. Only the bound on $|e-\epsilon|$ in hypothesis matters. ■

Can both perturbations be applied to Wiener's Theorem concurrently? The result of Blömer and May allows a multiplicative perturbation: $\epsilon d \equiv 1 \bmod \phi(n)$ becomes $\epsilon\delta \equiv c \bmod \phi(n)$. Our result allows an additive perturbation: if $|e-\epsilon|$ is not too large then knowing e is sufficient to efficiently yield δ. This is complementary to Theorem 7.1.1 in which a multiplicative perturbation of ϵ is known. Combining Theorems 7.1.1 and 7.1.3 together yields a situation in which e is the known additive approximation to ϵ, s.t. $\epsilon\delta \equiv c \bmod \phi(n)$.

This combination of theorems results in the following Linear Wiener's Theorem. Next, we build a backdoored key generator, G_1', that is based on this theorem.

Theorem 7.1.5 (Linear Wiener's Theorem) *Suppose that $\gamma \leq 1$, $p-q \geq \gamma n^{1/2}$, $\alpha \leq 1/4$, $\ell_1 < \frac{1}{3}$ and ℓ_2 s.t. $\ell_1^2(\ell_2+3) \leq \frac{1}{3}$. Given (n,e) related to (n,ϵ), with concealed ϵ, that satisfies*

$$\epsilon\delta - \kappa\phi(n) = c \tag{7.6}$$

such that

$$\begin{aligned} 0 < \quad & \delta & < \ell_1 \, n^{\alpha} \\ |c| \quad & \leq & \gamma n^{-3/4}\epsilon\delta \\ |e-\epsilon| \quad & < & \ell_2 \, n^{1-2\alpha}, \end{aligned}$$

then n can be factored efficiently.

Proof: Only the first component of Theorem 7.1.1's proof (cf. *Proof outline*) needs modifications, that is, that the parameters δ, κ can be found among the convergents of the continued fraction expansion of e/n. This can be done via Inequality 7.5, obtained in the proof of Theorem 7.1.3. For this inequality to hold, it remains to show that

$$\left|\frac{\epsilon}{n} - \frac{\kappa}{\delta}\right| < \frac{3\ell_1}{\delta n^{1/4}}$$

i.e.

$$3\ell_1 n^{3/4} > |\epsilon\delta - \kappa n|$$

so by Equation 7.6 and the definition of $\phi(n) = n - p - q + 1$

$$\begin{aligned} 3\ell_1 n^{3/4} &> \left|(\cancel{\kappa\phi(n)} + c) - \kappa(\cancel{\phi(n)} - p - q + 1)\right| \\ &= |\kappa(p + q - 1) + c|. \qquad (7.7) \end{aligned}$$

The latter can be shown as follows. Almost exactly as in the first component of Theorem 7.1.1's proof,

$$\begin{aligned} \kappa(p + q - 1) + c &\leq \frac{6}{4}\frac{\epsilon\delta}{\phi(n)} n^{1/2} + \gamma n^{-3/4}\epsilon\delta \qquad (7.8) \\ &\leq \frac{3}{2}\delta n^{1/2} + \delta n^{1/4} \\ &\leq 3\delta n^{1/2} \qquad (7.9) \end{aligned}$$

where Inequality 7.8 comes from $p + q \leq 2n^{1/2}$, the bound on c in hypothesis, and the following approximation of κ. This variable is isolated in Equation 7.6

$$\kappa = \frac{\epsilon\delta - c}{\phi(n)}$$

and approximated with $|c| \leq \gamma n^{-3/4}\epsilon\delta$ and $n \geq 2^3$, yielding $|c| \leq 2^{-9/4}\epsilon\delta$ and

$$\kappa \leq \frac{\epsilon\delta(1 - 2^{-9/4})}{\phi(n)} < \frac{3}{4}\frac{\epsilon\delta}{\phi(n)}.$$

Inequality 7.9 also holds for $n \geq 2^4$. That bound on n comes from that we can assume that $n \geq (8/c)^4$, otherwise $p - q \in \mathcal{O}(n^{1/4})$ in which case Fermat's factorization algorithm succeeds in polynomial time.

Because of the bounds on δ and α, it holds that $\delta < \ell_1 n^{\alpha} \leq \ell_1 n^{1/4}$. This implies that the RHS of Inequation 7.9 is bounded above by

$$3\delta n^{1/2} < 3\ell_1 n^{3/4},$$

which is the bound of Inequation 7.7 that was to be shown. So the requirement of the theorem on continued fractions (e.g. [HW79, Theorem 184]) is fulfilled. ■

Theorem 7.1.6 (Linear Wiener's Theorem, updated) *The statement of Theorem 7.1.5 holds for $\epsilon \in \mathbb{Z}_{\phi(n)}$ and $e \in \mathbb{Z}$ in general.*

Proof: Use Theorem 7.1.2 instead of Theorem 7.1.1; use Theorem 7.1.4 instead of Theorem 7.1.3. ■

We further generalize Theorem 6.3.1 by combining the Theorem 7.1.3 with the one that follows. Recall, from the preceding chapter, that Boneh, Durfree and Frankel showed several results allowing the recovery of the entire ϵ given a small δ, n and parts of ϵ (this notation swaps δ and ϵ w.r.t. the one used in [BDF98]). These results depend on one of Coppersmith [Cop97], which relates to Theorems 6.1.8 and 7.1.12.

Theorem 6.3.3 combined with our extension of Wiener's Theorem provides yet another contributed theorem. Given that δ is prime and small enough, one can derive (ϵ, δ) from (n, e), if an approximation e, to the public exponent ϵ, is known. The use of BDF's theorem is permitted by the fact that the approximation of ϵ reveals a number of its most significant bits. The parametrization with α was left out for clarity.

Theorem 7.1.7 (Extended Wiener combined with BDF) *Any (n, e) related to (n, ϵ), with concealed ϵ, that satisfies*

$$|e - \epsilon| < \frac{1}{4} n^{1/2}$$

such that

$$\delta \in [2^{k/2-3}, ..., 2^{k/2-2}]$$

efficiently yields the values of ϵ and δ.

Proof: Let $\alpha = \ell_1 = \ell_2 = 1/4$ in Theorem 7.1.3, yielding δ. If $\alpha = \ell_2 = 1/4$, then to satisfy the hypothesis of Theorem 7.1.3, $\ell_1 \leq 2/\sqrt{39} = \frac{1}{3}\sqrt{12/13}$, is required which includes $\ell_1 = 1/4$. From e, one knows $\epsilon\rceil^k$ and with $t = k/2 - 3$ in Theorem 6.3.3, efficiently derives ϵ. ■

Theorem 7.1.7 means that not only Wiener's small decryption exponents are weak, but also the encryption exponents that correspond to an approximation of a small exponent.

Theorem 7.1.8 (Extended Wiener combined with BDF, updated) *The statement of Theorem 7.1.7 holds for $e \in \mathbb{Z}$ in general.*

Proof: Use Theorem 7.1.4 instead of Theorem 7.1.3. ■

7.1.3 Improvement of the time complexity

We proceed to show a basic strategy which provides a linear time complexity for RSA backdoored key generators. Throughout this chapter, $\pi_\beta : \mathbb{Z}_{\phi(n)} \to \mathbb{Z}_{\phi(n)}$ denotes an invertible one-way, pseudo-random permutation such that given β, computing π_β^{-1} is easy.

The key to the permutation is the fixed secret key β, which remains the same for all legitimate users. This is simply coherent with symmetric cryptography, although it was not the case for the Anderson-Kaliski backdoor (Subsection 6.1.1): the secret key A is picked randomly for each legitimate user (*Symmetry* paragraph).

Proposition 7.1.9 (Running time) *The expected running times of some backdoored key generators and of Algorithm G_0 can be made of the same order.*

algorithm	figure	table
HG-1	6–2	6–2
HG-2	6–3	6–2
PAP	6–5	6–4
PAP-2	6–10	6–7
PP	6–11	6–8
CS-2	6–16	6–11
CS-3	6–17	6–12

Table 7–1: Pre-existing backdoored RSA key generators that have linear complexity, but assuming that a given cryptographic function has negligible complexity.

algorithm	figure	table
G_1	7–2	7–3
G_1'	7–6	7–4
G_3	7–8	7–6
G_4	7–9	7–7

Table 7–2: New backdoored RSA key generators that have linear complexity, but assuming that a π_β has negligible complexity in the first three cases only. Algorithm G_4 is the only one not needing this assumption.

Algorithm G_1 is the simplest backdoored key generator that satisfies Proposition 7.1.9, as follows. It is based on Theorem 7.1.1, unmodified except from the small update of Theorem 7.1.2. Because it relies on a multiplicative perturbation of a weak exponent, $\delta^{-1} \bmod \phi(n)$, it is called a *multiplicative* backdoored key generator.

In Step 5, the hypothesis $|c| \leq \gamma n^{-3/4}\epsilon\delta$ of Theorem 7.1.1 is replaced by the weaker condition $|c| \leq \gamma n^{1/4}\delta$, which is necessary because $\epsilon < n$. To insure that the original hypothesis is satisfied, a check is made in Step 8.

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$ is an assumed cryptographic permutation.
- γ is a constant that satisfies the hypotheses of Theorem 7.1.1: $\gamma \leq 1$.

Algorithm G_1 [Key generation]

```
 1: Pick rp $p, q$ of size $k$ s.t. $p - q \geq \gamma n^{1/2}$.
 2: Set $n = pq$.
 3: Pick a random $\delta \in \mathbb{Z}^*_{\phi(n)}$ s.t. $\delta < n^{1/4}/3$.
 4: do
 5:    Pick a random $c \in \mathbb{Z}_{\phi(n)}$ s.t. $c \leq \gamma n^{1/4}\delta$.
 6:    Set $\epsilon = c\delta^{-1} \bmod \phi(n)$.
 7:    Set $e = \pi_\beta(\epsilon)$.
 8: until $|c| \leq \gamma n^{-3/4}\epsilon\delta$ and $e \in \mathbb{Z}^*_{\phi(n)}$.
 9: Compute $d \equiv e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

Algorithm R_1 [Key retrieval]

```
1: Input of $(n, e)$.
2: Factor $n = pq$ from $(n, \pi_\beta^{-1}(e))$ [Theorem 7.1.2].
3: Compute $d \equiv e^{-1} \bmod \phi(n)$.
4: return $d$.
```

Figure 7–2: Our first symmetric backdoor for RSA, using a generalization of Wiener weak keys theorem of Blömer and May. The information on δ is embedded in e.

Essentials of proof:

Algorithm G_1: The function π_β is a permutation on the set $\mathbb{Z}_{\phi(n)}$, keyed with the secret β that "hides" the values of c. Its complexity is limited as to achieve the stated result. In Step 1, the RSA primes p and q, are set such that $p - q \geq \gamma n^{1/2}$, for some

$\gamma \leq 1$. The bounds on the choices of δ and c (which together determine ϵ), in Steps 3 and 5 are set so that (n, ϵ) allows efficient factorization of n via Theorem 7.1.2.

Algorithm R_1: The pair (n, e) is public and knowledge of β permits the computation of (n, ϵ). In Step 2, Theorem 7.1.2 yields p and q because the following conditions are satisfied, it holds that $\epsilon\delta \equiv c \bmod \phi(n)$ with $0 < \delta < \frac{1}{3}n^{1/4}$ and $|c| \leq \gamma n^{-3/4}\epsilon\delta$.

The running time of Algorithm G_0 is t_e as defined in Subsection 5.2.1. Consider two quantities: the probability, P, that the inverse modulo $\phi(n)$ of a random element of $\mathbb{Z}_{\phi(n)}$ exists; and the expected time, E, of an inversion modulo $\phi(n)$. We have found $t_e = E/P$ to be the total time of finding a random inverse in $\mathbb{Z}^*_{\phi(n)}$.

In comparison, the complexity of Algorithm G_1 for e is $t_e(G_1) = 2t_e$, that is, linear in the one of Algorithm G_0. Steps 3 takes time about $1/P$, because an invertible element is picked, but without being inverted. Steps 3 and 6 would together take t_e time if the latter step was not involved in the loop of Steps 4 to 8. Steps 5 to 7 are equivalent to picking a random e in $\mathbb{Z}_{\phi(n)}$, so the total complexity of Steps 4 to 9 is about t_e. The total, with Step 3, is bounded above by $t_e + t_e = 2t_e$.

Because here the running time is dominated by the greatest common divisor (gcd) checks and the inversion computations, the expected running times of G_0 and G_1 are both proportional to t_e. Also note that the generations of n of G_0 and G_1 are the same. Supposing that $T(\pi_\beta)$ is negligible, the expected running times of G_0 and G_1 are of the same order. ■

Algorithm properties. Algorithm G_1 is a generalization of CS-1. It uses Theorem 7.1.1 (unmodified except from the small update of Theorem 7.1.2) instead of Theorem 6.3.1.

Confidentiality. This property rests on the one-wayness of π_β.

Completeness. This property is based on Theorem 7.1.2.

Symmetry. Algorithm (G_1, R_1) is a symmetric backdoor with secret key β.

Cardinality. Key diversity is increased as compared to CS-1, as more pairs (δ, ϵ) are used for backdoor generation. There are

$$\begin{aligned} \mathcal{N}_{G_1,e} &\approx \#\{\delta\} \cdot \#\{c\} \\ &\approx \#\{\delta\} \cdot (2^{-3k/2} \cdot \#\{\delta\} \cdot \#\{\epsilon\}) \\ &\approx 2^{k/2} \cdot (2^{-3k/2} \cdot 2^{k/2} \cdot 2^{2k}) \\ &= 2^{3k/2} \end{aligned}$$

backdoored keys, if $f(c, \delta) = c\delta^{-1} \bmod \phi(n)$ is injective.

To show that $f(c, \delta)$ is approximately injective, one may show that the probability of collision is negligible. This is, for random (c, δ) and (c', δ'), the probability that

$$f(c, \delta) = f(c', \delta').$$

Given $\epsilon\delta = c$, the number of collisions (c', δ') for which $\epsilon\delta' = c'$ is the number of quotients $(\lfloor \frac{c}{D} \rfloor, \lfloor \frac{\delta}{D} \rfloor)$ for an integer D. So the number of collisions is approximately the minimum value of δ and c. Up to constants, the expected number of collisions is therefore

$$\begin{aligned}
E[(c',\delta')] &\approx \Pr[\delta < c]\cdot E[\delta] + \Pr[\delta > c]\cdot E[c] \\
&= \frac{\max\delta}{\max c}\max\delta + \left(1 - \frac{\max\delta}{\max c}\right)\max c \\
&= \frac{(\max\delta)^2}{\max c} + \max c - \max\delta \\
&\approx \frac{(n^{1/4})^2}{n^{1/2}} + n^{1/2} - n^{1/4} \\
&= 1 + n^{1/2} - n^{1/4},
\end{aligned}$$

using (up to constants) $\max\delta = n^{1/4}$ (Step 3) and $\max c = n^{-3/4}\epsilon\delta$ (Step 8) which expected value (up to constants) is $E[\max c] = n^{-3/4}\, n\, n^{1/4} = n^{1/2}$.

Then the probability of collision is (up to constants)

$$\begin{aligned}
\Pr[f(c,\delta) = f(c',\delta')] &= \frac{E[(c',\delta')]}{E[(c,\delta)]} \\
&\approx \frac{1 + n^{1/2} - n^{1/4}}{\max c\cdot\max\delta} \\
&\approx \frac{1 + n^{1/2} - n^{1/4}}{n^{1/4}\cdot n^{1/2}} \\
&\approx 2^{-k/2}
\end{aligned}$$

which is exponentially small, thus negligible. For reference purposes, this is written as a proposition.

Proposition 7.1.10 (Achievable cardinality of KSM I) *A set of backdoored keys that covers, in total for each run, $2^{\frac{3}{2}k}$ possible keys is generable in an expected running time of the same order as that of Algorithm G_0.*

Finally, in the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that $\mathcal{R}_{G_1} \approx 2^{-\frac{1}{2}k}$.

Distribution properties. The distribution of e is labeled "good", supposing that π_β is pseudo-random.

Entropy of embedding. The encrypted backdoor information is in the form of $e = \pi_\beta\left(c\delta^{-1} \bmod \phi(n)\right)$. Entropy is particularly improved from CS-1, because RSA exponents are nearly uniformly distributed, given Assumption 2.2.23. More formally, there are sufficient reasons to make the assumption that the properties of RSA imply that the distribution of $\pi_\beta(c\delta^{-1} \bmod \phi(n))$ is computationally indistinguishable from the one of $e \in_U \mathbb{Z}^*_{\phi(n)}$. [3]

Furthermore, $\epsilon = c\delta^{-1} \bmod \phi(n)$ makes the domain of π_β larger than if it were, as for CS-1, $\epsilon = \delta^{-1} \bmod \phi(n)$. If there were fewer δ^{-1} and smaller $|c|$, as in Figure 7–3, then ϵ could assume a lesser number of values in $\mathbb{Z}_{\phi(n)}$.

That there are about $k/2$ close to uniform values of δ^{-1}. The multiplication by the k bit random number that is c allows ϵ to possibly take more values in $\mathbb{Z}_{\phi(n)}$.

The entropy of G_1 is $\mathcal{H}_{G_1} = 1$, because of the nearly uniform distribution of the RSA public exponent, given Assumption 2.2.23 (the RSA private exponent in the assumption's statement). This distribution of δ, with the choice of parameters in the generation of ϵ, makes it so that ϵ is distributed on all of $\mathbb{Z}_{\phi(n)}$. Secondly, different

[3] This assumption is weaker than Assumption 2.2.22, by Theorem 2.2.24 (Subsection 2.2.8). Informally, this is assuming that the properties of RSA imply that the distribution of $\pi_\beta(\delta^{-1} \bmod \phi(n))$ is computationally indistinguishable from the one of $e \in_U \mathbb{Z}^*_{\phi(n)}$.

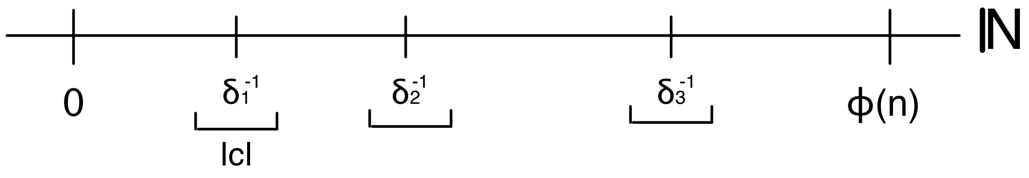


Figure 7–3: Entropy with few δ and small $|c|$. The possible values of $c \cdot \delta^{-1} \bmod \phi(n)$ reach a small number of values in the interval. That $c \cdot \delta^{-1} \bmod \phi(n)$ is larger at the right hand side of the $[0, \phi(n)]$ interval, so the sub-intervals are larger there as well, is not illustrated.

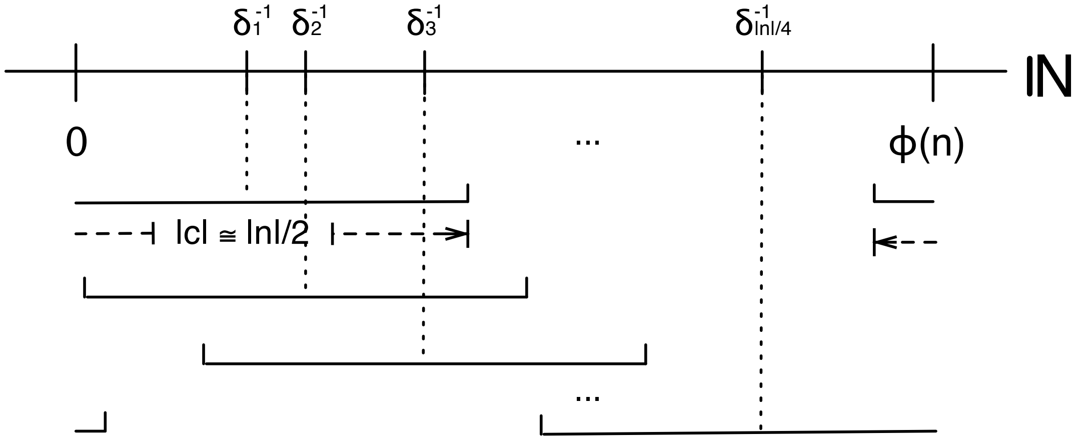


Figure 7–4: Entropy with $k/2$ values of δ and $|c| \approx k$. The possible values of $c \cdot \delta^{-1} \bmod \phi(n)$ reach more values in the interval.

samplings of δ yield mostly uncorrelated inverses, given Assumption 2.2.17′. This assumption is that different values of δ^{-1} are distributed in a *well pre-shuffled* way, despite that the values of δ are restricted to $k/2$ bits.

Considering all generable keys as in Figure 7–4, because the theorem of Blömer and May allows for $\epsilon\delta = c \neq 1$, we gain an expected additional k random bits per δ. This is illustrated in Figure 7–5. On the first line, G_1 generates about $k/2$ bits

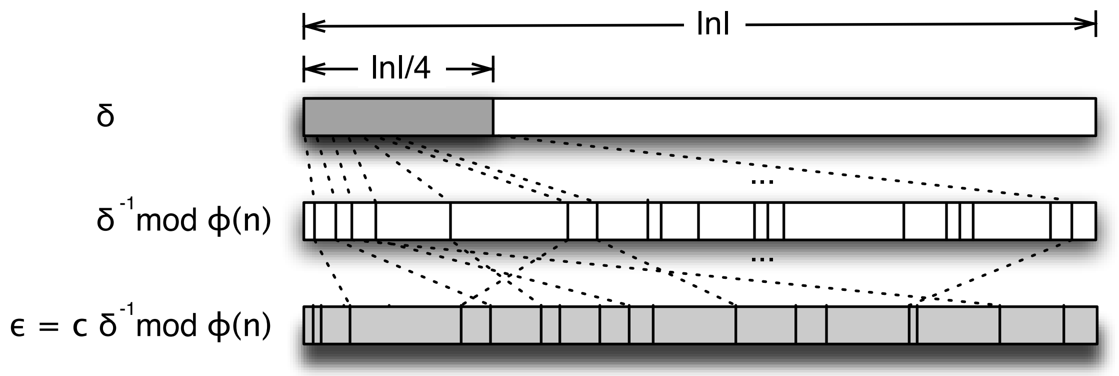


Figure 7–5: Entropy of G_1.

of δ. The shadowed bits are bits of backdoor information. On the second line, these bits are uniformly spread into $2k$ bits via the well pre-shuffled distribution of the RSA private exponent (given Assumption 2.2.17′). On the third line, for fixed δ and c, the value of ϵ is a noisy copy of the one of Figure 6–15: there are many more possibilities.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. Proposition 7.1.9 states that the complexity is proportional to the one of G_0. Overall:

$$\begin{aligned} T_n(G_1) &= t_n \\ T_e(G_1) &\approx t_e + T(\pi_\beta). \end{aligned}$$

These complexities are linear in the complexity of G_0. Assuming that $F = \pi_\beta$ is negligible with respect to t_e, Algorithm G_1 is rated “good”. Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_1 is rated “good / F-relative”.

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

The results in *Entropy* depend on Assumption 2.2.23.

Simplicity. As for CS-1, the algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			yes: one-way π_β	good
completeness			yes: Theorem 7.1.1	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_1}$	$2^{-\frac{1}{2}k}$	good
		$\mathcal{C}_{G_1}$	0, pseudo-random π_β	good
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	1, given Assumption 2.2.23	good
	GKR		yes	good
	side-channels	complexity	$T_n(G_1) = t_n$ $T_e(G_1) \approx t_e + T(\pi_\beta)$	good / F-rel.
		memory	no	good
computational assumptions			one-way, pseudo-random π_β, Assumption 2.2.23	average
simplicity			LS	good

Table 7–3: Properties of the Algorithm G_1 of Figure 7–2.

Discussion. Algorithm G_1 has greater diversity than the CS-1 because it is a noisy copy of that previous algorithm. Accordingly, the parameter c may be called a perturbation. The entropy of the keys and the difficulty of distinguishing them from honest ones are increased. Therefore G_1 increases indistinguishability while not allowing any loss of number of generable keys, and in fact, producing exponentially many more.

To proceed onto showing that a greater cardinality of KSM is achievable, improving Proposition 7.1.10, two complementary simple algorithms that match the running time of (G_1, R_1) are shown. After studying (G_2, R_2), one may wonder why it is useful, because it is less efficient and generates less keys. Nevertheless, it gives intuition toward our best result (and this is its only purpose). Secondly, (G_3, R_3), in turn gives intuition for (G_4, R_4), which is finally used by the proof of Proposition 7.1.14, stating that greater cardinality of KSM is achievable.

However, we first show G'_1, a generalization of G_1 that produces a perfect cardinality. The reasons for the study of the precedingly mentioned three future algorithms are highlighted in the *Discussion* which follows the presentation of G'_1.

A generalization of Algorithm G_1. A simple backdoored key generator can be derived from Theorem 7.1.5, as shown in Figure 7–6. Algorithm G'_1 is therefore a generalization of G_1. This is useful for comparison purposes, with other backdoored key generators based on more complicated results. For most choices of parameters, we show a backdoored key generator and retrieval algorithm pair, (G_4, R_4), which requires a more complicated analysis, but yields better results than (G'_1, R'_1).

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \to \{0,1\}^{2k}$ is an assumed cryptographic permutation.
- $\gamma, \alpha, \ell_1, \ell_2$ are constants that satisfy the hypotheses of Theorem 7.1.5.

Algorithm G'_1 [Key generation]

```
 1: Pick rp $p, q$ of size $k$ s.t. $p - q \geq \gamma n^{1/2}$.
 2: Set $n = pq$.
 3: Pick a random $\delta \in \mathbb{Z}^*_{\phi(n)}$ s.t. $\delta < \ell_1 n^\alpha$.
 4: do
 5:     Pick a random $a \in \mathbb{Z}_{\phi(n)}$ s.t. $a < \ell_2 n^{1-2\alpha}$.
 6:     Pick a random $c \in \mathbb{Z}_{\phi(n)}$ s.t. $c \leq \gamma n^{1/4}\delta$.
 7:     Set $\epsilon \equiv c\delta^{-1} + a \bmod \phi(n)$.
 8:     Set $e = \pi_\beta(\epsilon)$.
 9: until $|c| \leq \gamma n^{-3/4}\epsilon\delta$ and $e \in \mathbb{Z}^*_{\phi(n)}$.
10: Compute $d \equiv e^{-1} \bmod \phi(n)$.
11: return $(p, q, d, e)$.
```

Algorithm R'_1 [Key retrieval]

```
1: Input of $(n, e)$.
2: Factor $n = pq$ from $(n, \pi_\beta^{-1}(e))$ [Theorem 7.1.6].
3: Compute $d \equiv e^{-1} \bmod \phi(n)$.
4: return $d$.
```

Figure 7–6: A symmetric backdoor for RSA based on a linear extension of Wiener's Theorem. The information on δ is embedded in e.

Algorithm properties. The following adjustment is the same as for G_1. In Step 6, the hypothesis $|c| \leq \gamma n^{-3/4}\epsilon\delta$ of Theorem 7.1.5 is replaced by the weaker condition $|c| \leq \gamma n^{1/4}\delta$, which is necessary because $\epsilon < n$. To insure that the original hypothesis is satisfied, a check is made in Step 9.

Confidentiality. This property rests on the one-wayness of π_β.

Completeness. At Step 2 of R'_1, it holds that $(n, \pi_\beta^{-1}(e)) = (n, \epsilon)$ is such that $|\epsilon - \epsilon_o| < \ell_2\, n^{1-2\alpha}$ for $\epsilon_o \equiv c\delta^{-1} \bmod \phi(n)$. For some integer κ, it holds that $\epsilon_o\delta - \kappa\phi(n) = c$ with the given bounds on δ, c. The conditions of Theorem 7.1.6 are thus satisfied with $\epsilon = \epsilon_o$ and $e = \epsilon$. Completeness is shown by this theorem.

Symmetry. Algorithm (G'_1, R'_1) is a symmetric backdoor with secret key β.

Cardinality. Without counting collisions in the generation of

$$\epsilon \;\equiv\; c\delta^{-1} + a \bmod \phi(n) \tag{7.10}$$

the number of generable keys would be $2^{(\alpha+1-2\alpha+1/4+\alpha)2k} = 2^{5/2k}$, which cannot be the case, as it exceeds 2^{2k}. Consider instead both terms of Equation 7.10 separately, since the second term only affects the lower $(2-4\alpha)k$ bits and thus does not affect the remaining upper $4\alpha k$ bits.

The first term, $c\delta^{-1}$, is composed of $|c| = (1/4+\alpha)2k$ and $|\delta| = 2\alpha k$ bits. As in the analysis of Algorithm G1, the $(1/4+2\alpha)2k$ bits of $c\delta^{-1}$ are spread nearly uniformly amongst the $2k$ bits of ϵ (Figure 7–5). Therefore, given Assumption 2.2.17′, in the fraction of ϵ unaffected by the addition of a, that is, its $4\alpha k$ upper bits, one expects that $c\delta^{-1}$ contributes the following number of random bits.

$$\begin{aligned}\#\text{free bits from } c\delta^{-1} &= (\text{average \# of free bits per bit}) \cdot (\#\text{bits}) \\ &= (1/4+2\alpha)\cdot 4\alpha k\end{aligned}$$

The second term, a, overwrites the randomness from $c\delta^{-1}$ in the lower bits. There are $(2-4\alpha)k$ such lower bits. This accounts for a total of

$$\lg \mathcal{N}_{G'_1,e}(\alpha) \;=\; (1/4+2\alpha)4\alpha k + (2-4\alpha)k = (8\alpha^2 - 3\alpha + 2)k.$$

In the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that $\mathcal{R}_{G'_1} = 2^{(8\alpha^2-3\alpha)k}$.

Bounds on $\mathcal{R}_{G'_1}$ can be established. Because

$$\frac{d}{d\alpha} \lg \mathcal{R}_{G'_1}(\alpha) = \frac{d}{d\alpha}(8\alpha^2 - 3\alpha) = 16\alpha - 3,$$

the ratio of keys is minimized when $\alpha = 3/16$ with

$$\mathcal{R}_{G'_1} \approx 2^{(8(3/16)^2-9/16)k} = 2^{-\frac{9}{32}k}.$$

If $\alpha = 1/4$, one expects a ratio of

$$\mathcal{R}_{G'_1} \approx 2^{(8(1/4)^2-3/4)k} = 2^{-\frac{1}{4}k}$$

and if $\alpha = 0$, one expects a ratio of

$$\mathcal{R}_{G'_1} \approx 2^{0k} = 1.$$

Therefore, one expects

$$2^{-\frac{9}{32}k} \le \mathcal{R}_{G'_1} \le 1.$$

Distribution properties. The distribution of e is labeled “good”, supposing that π_β is pseudo-random.

Entropy of embedding. The same as for the preceding algorithm holds, with a minor modification to Theorem 2.2.24. Instead of producing another theorem, this modification is highlighted. Distribution 2.8 changes to

$$(p, q, D_1, ..., D_{poly(k)}) \quad \text{where} \quad d_i \equiv \pi_\beta(c \cdot e_i^{-1} \bmod \phi(n) + a) \qquad (7.11)$$

where c and a satisfy the hypotheses of Theorem 7.1.5.

Recall the probability distributions that were compared in the original theorem. Distribution 2.7 is the honest distribution and Distribution 2.6 is the one of objects of the form $\pi_\beta\left(e^{-1} \bmod \phi(n)\right)$.

$$\begin{aligned}
&(p, q, D_1, ..., D_{poly(k)}) &\text{where}\quad & D_i \equiv \pi_\beta\left(e^{-1} \bmod \phi(n)\right) && (2.6)\\
& &\text{and}\quad & \gcd(D_i, \phi(n)) = 1 &&\\
&(p, q, d'_1, ..., d'_{poly(k)}) &\text{where}\quad & d'_i \in_U \mathbb{Z}^*_{\phi(n)} && (2.7)
\end{aligned}$$

Suppose that $\mathcal{D}$ distinguishes the two former distributions with probability $1/2+\varepsilon$. Let x be sampled uniformly from Distributions 2.6 or 2.7, and appropriately distributed c and a. Then Equation 2.9 changes to

$$\pi_\beta\left(c \cdot \pi_\beta^{-1}(x) + a\right) \in \begin{cases} \text{Distribution 7.11} & \text{if } x \in \text{Distribution 2.6,} \\ \text{Distribution 2.7} & \text{if } x \in \text{Distribution 2.7.} \end{cases}$$

A useful assumption is one by which Distribution 7.11 is computationally indistinguishable from Distribution 2.7. This assumption is weaker than Assumption 2.2.23. This holds because the latter indistinguishability assumption implies the former. To show this, an argument similar to the one used in the proof of Theorem 2.2.24 can be used. In other words, multiplying by c and adding a, both appropriately distributed, transforms Distribution 2.6 into Distribution 7.11 and leaves Distribution 2.7 unchanged. Therefore

$$\mathcal{D}'(x) = \mathcal{D} \circ \pi_\beta\left(c \cdot \pi_\beta^{-1}(x) + a\right)$$

distinguishes Distributions 2.6 and 2.7 with probability $1/2 + \varepsilon$.

Denote this weaker assumption as Assumption 2.2.23′.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. Proposition 7.1.9 applies: the complexity is proportional to the one of G_0. Overall:

$$\begin{aligned} T_n(G_1') &= t_n \\ T_e(G_1') &\approx t_e + T(\pi_\beta). \end{aligned}$$

These complexities are linear in the complexity of G_0. Assuming that $F = \pi_\beta$ is negligible with respect to t_e, Algorithm G_1' is rated “good”. Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_1' is rated “good / F-relative”.

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

The results in *Cardinality* depend on Assumption 2.2.17′. Those in *Entropy* depend on Assumption 2.2.23′, which already supposed 2.2.17′.

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			yes: one-way π_β	good
completeness			yes: Theorem 7.1.6	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_1'}$	$2^{-\frac{9}{32}k} \leq \mathcal{R}_{G_1'} \leq 1$	good
		$\mathcal{C}_{G_1'}$	0, pseudo-random π_β	good
		$\mathcal{V}_{G_1'}$	1	good
		$\mathcal{H}_{G_1'}$	1, given Assumption 2.2.23′	good
	GKR		yes	good
	side-channels	complexity	$T_n(G_1) = t_n$ $T_e(G_1) \approx t_e + T(\pi_\beta)$	good / F-rel.
		memory	no	good
computational assumptions			one-way, pseudo-random π_β, Assumption 2.2.17′, Assumption 2.2.23′	average
simplicity			LS	good

Table 7–4: Properties of Algorithm G_1' of Figure 7–6.

Discussion on the cardinality. When $\alpha = 0$, it holds that $\mathcal{R}_{G_1'} = 1$. In this case, there are few choices of δ, therefore the keys are generated and retrieved from the additive perturbation in Theorem 7.1.5. Because of the pseudo-randomness of π_β, the correlation with δ cannot computationally be detected even if all keys used the same δ. However, this implies that the security of G_1' is entirely based on the pseudo-randomness of π_β. We will show Algorithm G_4 which removes the pseudo-randomness assumption and instead uses the theorem's power more fully, at the cost of a lesser cardinality. However, in order to build intuition, we first show two intermediary algorithms, G_2 and G_3.

7.1.4 Improvement of the diversity without pseudo-randomness

We proceed to show a succession of three consecutively improved algorithms that provide an improved diversity, w.r.t. the existing algorithms and G_1.

Algorithm G_2 increases entropy w.r.t. G_1

We show an algorithm that increases the entropy of the backdoored keys w.r.t. G_1. The cardinality will be diminished, but the next algorithm will remedy to this, while preserving the acquired entropy.

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$ is an assumed cryptographic permutation.

```
Algorithm G_2 [Key generation]
 1: Pick rp $p, q$ of size $k$ s.t. $q < p < 2q$.
 2: Set $n = pq$.
 3: Pick a rp $\delta \in [2^{k/2-3},...,2^{k/2-2}] \cap \mathbb{Z}^*_{\phi(n)}$.
 4: Set $\epsilon \equiv \delta^{-1} \bmod \phi(n)$.
 5: repeat
 6:     Pick random $a$ s.t. $|a| < n^{1/2}/4$; $0 < \epsilon + a < n$.
 7:     Set $e \equiv \pi_\beta(\epsilon + a)$.
 8: until $\gcd(e, \phi(n)) = 1$.
 9: Compute $d \equiv e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

```
Algorithm R_2 [Key retrieval]
1: Input of $(n, e)$.
2: Compute $\delta, \epsilon$ from $(n, \pi_\beta^{-1}(e))$ [Theorem 7.1.8].
3: Factor $n = pq$ from $\epsilon$ and $\delta$.
4: Compute $d \equiv e^{-1} \bmod \phi(n)$.
5: return $d$.
```

Figure 7–7: Our second symmetric backdoor for RSA, using another variant of Wiener, derived from combining an extension of Wiener with a theorem from BDF. The information on δ is embedded in e.

Algorithm properties.

Confidentiality. This property rests on the one-wayness of π_β.

Completeness. This property is based on Theorem 7.1.8, supposing that the permutation, π_β is efficiently invertible, knowing β. The theorem is satified as follows. The bounds on the choices of δ (hence affecting ϵ) and a, in Steps 3 and 6 are set so that the pair $(n, \epsilon + a)$, allows efficient factorization of n via Theorem 7.1.8. From the public (n, e) and knowledge of β, one finds $(n, \epsilon + a)$. In Step 2 of R_2, Theorem 7.1.8 yields (ϵ, δ), because

$$|a| = |\pi_\beta^{-1}(e) - \epsilon| < \frac{1}{4} n^{1/2}$$

and

$$\delta \in [2^{k/2-3}, ..., 2^{k/2-2}]$$

is prime.

Symmetry. Algorithm (G_2, R_2) is a symmetric backdoor with secret key β.

Cardinality. Algorithm G_1 can generate any of Wiener's weak $\delta < \frac{1}{3}n^{1/4}$, whereas Algorithm G_2 is limited to prime

$$\delta < \frac{1}{3}\sqrt{12/13}n^{1/4}.$$

Therefore, the latter can generate about $2^{\frac{3}{2}k}/\ln(n)$ backdoored keys. This $1/\ln(n)$ factor is going to be patched in order to yield $2^{\frac{3}{2}k}$ backdoored keys in the upcoming algorithms (G_3, R_3) and (G_4, R_4). (The $|\epsilon + a| < 2^{2k}$ condition in Step 6 does affects the cardinality by less than a constant, therefore insignificantly.)

In the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that $\mathcal{R}_{G_2} = 2^{-\frac{1}{2}k}/\ln(n)$.

Distribution properties. The distribution of e is labeled "good", supposing that π_β is pseudo-random.

Entropy of embedding. The same as for the preceding two algorithms holds, with a minor modification to Theorem 2.2.24, comparable to the one for Algorithm G_1'. Denote the resulting modified assumption on distribution properties of RSA as Assumption 2.2.23$^{(2)}$.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. The computation of n is the same as in G_0.

$$T_n(G_2) \quad = \quad t_n$$

As for $T_e(G_2)$, it would be comparable to $T_e(G_1)$ if the former did not pick a random *prime* δ (Step 3), although it is only of half the size of p or q. This accounts for an additional complexity of $t_{\sqrt{n}}$, just as it was the case for Algorithm CS-2 (Figure 6–16). Also, Proposition 7.1.9 does not apply exactly, because π_β is used within the loop for the generation of e. Overall:

$$T_e(G_2) \quad = \quad t_{\sqrt{n}} + t_e + t_e \cdot T(\pi_\beta) \approx t_{\sqrt{n}} + t_e \cdot T(\pi_\beta).$$

These complexities are linear in the complexity of G_0 and $t_{\sqrt{n}}$ has significantly smaller complexity than RSA (the one of G_0) as a whole, of which t_e is a significant term. Assuming that $F = \pi_\beta$ is negligible with respect to t_e, Algorithm G_2 is rated "good". Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_2 is rated "good / F-relative".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

The results in *Entropy* depend on Assumption 2.2.23$^{(2)}$.

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			yes: one-way π_β	good
completeness			yes: Theorem 7.1.8	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_2}$	$2^{-\frac{1}{2}k}/\ln(n)$	good
		$\mathcal{C}_{G_2}$	0, pseudo-random π_β	good
		$\mathcal{V}_{G_2}$	1	good
		$\mathcal{H}_{G_2}$	$1,$given Assumption 2.2.23$^{(2)}$	good
	GKR		yes	good
	side channels	complexity	$T_n(G_2) = t_n$ $T_e(G_2) \approx t_{\sqrt{n}} + t_e \cdot T(\pi_\beta)$	good / F-rel.
		memory	no	good
computational assumptions			one-way, pseudo-random π_β, Assumption 2.2.23$^{(2)}$	average
simplicity			LS	good

Table 7–5: Properties of Algorithm G_2 of Figure 7–7.

Discussion. Algorithm G_2 is complementary to G_1: using the theorem with approximate ϵ (Th. 7.1.8), instead of the theorem without approximation (Th. 7.1.1), yields more entropy. But G_2 puts more restrictions on δ, hence decreasing the number of keys. G_3 deals with this issue.

Achieving the non-primality of δ by G_3

To pursue the insight of G_2, consider G_3 which directly uses Theorem 6.3.4, with $t = k/2 - 3$ [4] . Algorithm G_3 removes the restrictions on δ, hence restoring cardinality while preserving entropy. It is comparable to CS-3, with a complexity given by Proposition 7.1.9.

PARAMETERS:
- β is a fixed secret key.
- $\pi_\beta : \{0,1\}^{2k} \rightarrow \{0,1\}^{2k}$ is an assumed cryptographic permutation.

```
Algorithm $G_3$ [Key generation]
 1: Pick rp $p, q$ of size $= k$ s.t. $q < p < 2q$.
 2: Set $n = pq$.
 3: Pick a random $\delta \in \mathbb{Z}^*_{\phi(n)}$ such that $\delta < n^{1/4}/4$.
 4: Set $\epsilon \equiv \delta^{-1} \bmod \phi(n)$.
 5: repeat
 6:     Pick rd. $c$ s.t. $|c| < n^{1/4}/4$; $0 < \epsilon + c2^{k/2} < n$.
 7:     Set $e = \pi_\beta(\epsilon + c2^{k/2})$.
 8: until $\gcd(e, \phi(n)) = 1$.
 9: Compute $d \equiv e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

```
Algorithm $R_3$ [Key retrieval]
1: Input of $(n, e)$.
2: Set $\epsilon\rfloor_{k/2} = \pi_\beta^{-1}(e) \bmod 2^{k/2}$.
3: Set $\epsilon\rceil^{k/2-3} = \pi_\beta^{-1}(e)\rceil^{k/2-3}$.
4: Find $\delta$ from $(n, \pi_\beta^{-1}(e))$ [Theorem 7.1.4].
5: Find $p$, $q$ from $(n, \delta)$, $\epsilon\rfloor_{k/2}, \epsilon\rceil^{k/2-3}$ [Th. 6.3.4].
6: Compute $d \equiv e^{-1} \bmod \phi(n)$.
7: return $d$.
```

Figure 7–8: Our third symmetric backdoor for RSA, using an extension of Wiener's Theorem and another theorem from BDF. The information on δ is embedded in e.

Algorithm properties. In an implementation of (G_3, R_3), it would be simpler to apply π_β on the $k/2 - 3$ most significative bits of e only.

Confidentiality. This property rests on the one-wayness of π_β.

Completeness. This property is based on Theorem 7.1.3, supposing that the permutation, π_β is efficiently invertible, knowing β. The theorem is satified as follows.

[4] From the preceding chapter, Theorem 6.3.4 is useful for intuition. Our Theorem 7.1.13 is its generalization.

Let π_β be a permutation on $[-n^{1/4}/4+1, ..., n^{1/4}/4-1]$, with secret key β that hides c. In Step 4 of R_3, Theorem 7.1.4 yields δ because the following are satisfied: with $\alpha = \ell_1 = \ell_2 = 1/4$, it holds that

$$\begin{aligned} |\pi_\beta^{-1}(e) - \epsilon| &= |\pi_\beta(c)|\, 2^{k/2} \\ &< n^{1/2}/4 \end{aligned}$$

and that $\delta < n^{1/4}/4$.

Symmetry. Algorithm (G_3, R_3) is a symmetric backdoor with secret key β.

Cardinality. Even though Theorem 7.1.3 (equivalently, Theorem 7.1.4) allows an approximation of size about $n^{1/2}$, Theorem 6.3.4 requires that the $k/2$ upper bits of ϵ as well as its $k/2$ lower bits be retrieved. Therefore, no perturbation should affect the $k/2$ lower bits. Since Theorem 7.1.3's approximation is additive (therefore only the k lower bits are perturbable), there remains $k/2$ bits that are perturb-able. Accounting for δ being of $k/2$ bits, the cardinality of KMS sums up to k bits.

In the ratio of the cardinalities of KSM and KS, the $\#\{n\}$ terms can be canceled so that $\mathcal{R}_{G_3} = 2^{-k}$.

Distribution properties. The distribution of e is labeled "good", supposing that π_β is pseudo-random.

Entropy of embedding. The same as for the preceding three algorithms holds, with a minor modification to Theorem 2.2.24, comparable to the one for Algorithm G_1'. Denote the resulting modified assumption on distribution properties of RSA as Assumption 2.2.23$^{(3)}$.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. In part similarly to the preceding algorithm, Proposition 7.1.9 does not apply exactly, because π_β is used within the loop for the generation of e. Overall, the generation of n is the same and so is the one of e, but without the generation of a prime.

$$\begin{aligned} T_n(G_3) &= t_n \\ T_e(G_3) &= t_e + t_e \cdot T(\pi_\beta) \approx t_e \cdot T(\pi_\beta). \end{aligned}$$

These complexities are linear in the complexity of G_0. Assuming that $F = \pi_\beta$ is negligible with respect to t_e, Algorithm G_3 is rated "good". Overall, the complexity is less than quadratic in the complexity of G_0, while the complexity of F is not given explicitly. Therefore, G_3 is rated "good / F-relative".

Memory. No use of additional memory is made.

Computational assumptions. The algorithm uses a family of pseudo-random functions, π_β, which is not explicitly provided.

The results in *Entropy* depend on Assumption 2.2.23$^{(3)}$.

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

Discussion. G_3 generalizes G_2. It removes the requirement for δ to be prime by using Theorems 7.1.4 and 6.3.4, instead of 7.1.8. Because there are less restrictions

confidentiality			yes: one-way π_β	good
completeness			yes: Theorems 7.1.4 and 6.3.4	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_3}$	2^{-k}	poor
		$\mathcal{C}_{G_3}$	0, pseudo-random π_β	good
		$\mathcal{V}_{G_3}$	1	good
		$\mathcal{H}_{G_3}$	$1,$ given Assumption 2.2.23$^{(3)}$	good
	GKR		yes	good
	side channels	complexity	$T_n(G_3) = t_n$ $T_e(G_3) \approx t_e \cdot T(\pi_\beta)$	good / F-rel.
		memory	no	good
computational assumptions			one-way, pseudo-random π_β, Assumption 2.2.23$^{(3)}$	average
simplicity			LS	good

Table 7–6: Properties of Algorithm G_3 of Figure 7–8.

on δ, there are more bits of backdoor information, which, as for Algorithm CS-1, are uniformly distributed. Thus, this increases key entropy but decreases cardinality.

We have three successive improvements on Algorithms CS-1 and CS-3. Next we must specify π_β.

The generation of $2^{2(1-\alpha)k}$ backdoored keys by G_4

The fourth algorithm (G_4, R_4) is entirely based on the structure of RSA. No other one-way function is involved. The designer uses a secret key β to achieve confidentiality. Its completeness rests on an additive generalization of Wiener's Theorem, given in the same paper (Theorem 7.1.3), as well as on one of Coppersmith's factorization method from approximate values of p (Theorem 7.1.12).

The three following theorems are rounded up by Theorem 7.1.13, which proof is a generalization of the one of Theorem 6.3.4.

Theorem 7.1.11 ([Red96], Theorem 3.16) *A necessary and sufficient condition for* $y^2 \equiv b \pmod{\beta}$*, to be solvable for y, where* $\gcd(b, \beta) = 1$*, is that b be a quadratic residue of all odd prime divisors of* β *and that if* $2||\beta$*, then b is odd, if* $4||\beta$*, then* $b \equiv 1 \pmod 4$*, and if* $8|\beta$*, then* $b \equiv 1 \pmod 8$*.*

If solvable, square roots modulo a prime p are found in expected $\mathcal{O}(\lg^3 p)$ by Cipolla's Algorithm [BS96, Section 7.2]. The Chinese Remainder Theorem applies for composite β with known factorization. A valuable succession of results is the proof [RS86] that the $2k/3$ most significant bits of p suffice to factor n efficiently and an improvement [Cop96] which reduces the number of bits to $k/2$. Thus recall, from the previous chapter, how an approximate p leads to factoring of n if the additive error is on its lower-half bits via Corollary 6.1.8.

Corollary 6.1.8 yields Theorem 7.1.12. The latter is used to generalize Theorem 6.3.4 and yield Theorem 7.1.13, using that in Theorem 7.1.12, it is not required that β be a power of 2. Recall a corollary to Coppersmith's Theorem (Theorem 6.1.7) and its proof, as it is a simple application of this theorem.

Theorem 7.1.12 ([BDF98], Corollary 2.2) *Given $\beta \geq 2^{k/2}$ and $p_0 \equiv p \bmod \beta$, it is possible to factor $n = pq$ in time polynomial in k, denoted by $T(k)$.*

proof: The proof of Theorem 6.1.7 rests on the LLL lattice basis reduction. It yields Theorem 7.1.12, by taking $p_o \equiv p \bmod \beta$, $q_o \equiv n/p_o \bmod \beta \equiv q \bmod \beta$ and letting

$$f(x, y) = (\beta x + p_o)(\beta y + q_o) - n.$$

It is assumed that $\gcd(p, \beta) = 1$, otherwise, factoring n is trivial.

The solution sought to $f(x, y)$ is (x_o, y_o) such that $0 \leq x_o < X = 2^{k+1}/\beta$ because of the limitation on the size of the primes (and similarly for y_o). Because the gcd of the coefficients of f is β, we instead use Theorem 6.1.7 on $g(x, y) = f(x, y)/\beta$. Determine the largest coefficient, D, of $\tilde{g}(x, y) = g(Xx, Yy)$.

$$\tilde{g}(x, y) \;=\; \frac{1}{\beta} f(Xx, Yy) = \beta XYxy + Xq_o x + Yq_o y + \frac{p_o q_o - n}{\beta}$$

Therefore, using $X = Y = 2^{k+1}/\beta$:

$$D = |\beta XY| = \beta \left(2^{k+1}/\beta\right)^2 = 2^{2k+2}/\beta.$$

The maximum degree of a variable is $\delta = 1$. To use the theorem, it is thus required that $XY < D^{2/(3\delta)} = D^{2/3}$. Equivalently:

$$\begin{aligned} 2^{2k+2}/\beta^2 &< (2^{2k+2}/\beta)^{2/3} \\ \text{and} \quad 2^{(2k+2)/3} &< \beta^{4/3} \end{aligned}$$

which is satisfied when $\beta > 2^{(k+1)/2}$. By exhaustive search on the first two bits of x_o, y_o, this is reduced to $\beta \geq 2^{k/2}$. ■

The following algorithm yields the strongest result concerning the number of backdoored keys.

PARAMETERS:
- β is a fixed secret key.
- $\alpha \leq 1/4$, $\ell_1 < \frac{1}{3}$ and ℓ_2 s.t. $\ell_1^2(\ell_2 + 3) \leq \frac{1}{3}$ are constants that satisfy the hypotheses of Theorem 7.1.3.

```
Algorithm G_4 [RSA key generation]
 1: Pick rp $p, q$ of size $= k$ s.t. $q < p < 2q$.
 2: Set $n = pq$.
 3: Pick a random $\delta \in \mathbb{Z}^*_{\phi(n)}$ such that $\delta < \ell_1 n^\alpha$.
 4: Set $\epsilon \equiv \delta^{-1} \bmod \phi(n)$.
 5: repeat
 6:     Pick rd. $c$ s.t. $|c| < \frac{\ell_2}{\beta} n^{1-2\alpha}$; $0 < \epsilon + c\beta < n$.
 7:     Set $e = \epsilon + c\beta$.
 8: until $\gcd(e, \phi(n)) = 1$.
 9: Compute $d \equiv e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

```
Algorithm R_4 [Key retrieval]
1: Input of $(n, e)$.
2: Set $\epsilon_0 = e \bmod \beta$.
3: Compute $\kappa$, $\delta$ from $(n, e)$ [Theorem 7.1.4].
4: Set $a = [\epsilon_0\delta - \kappa(n+1) - 1]/(2\kappa)$ and $b = a^2 - n$.
5: Set $p_0$ to be a solution of $(x - a)^2 \equiv b \pmod{\beta}$
   [Theorem 7.1.11].
6: Compute $p$, $q$ from $\beta$, $p_0$ [Theorem 7.1.12].
7: Compute $d \equiv e^{-1} \bmod \phi(n)$.
8: return $d$.
```

Figure 7–9: Our fourth symmetric backdoor for RSA, using extended Wiener weak keys and an extended theorem of Boneh, Durfree and Frankel. The information on δ is embedded in e.

Algorithm properties. The algorithm uses a private key β, which is a modulo that is generally not a power of 2. Its value is picked, fixed and kept private by the inverter as an odd random number of known factorization [Kal03], of size $k/2$. Therefore, it satisfies the hypotheses of Theorem 7.1.12. The values of ℓ_1, ℓ_2, and $\alpha < 1/6$ satisfy the hypotheses of Theorem 7.1.3 [5] .

For Step 3 of R_4, the argument is the same as for Step 4 in R_3, except that Theorem 7.1.3 is used in its general, parameterized form. The correctness of the remaining

[5] In Step 7, $\pi_\beta(\epsilon) = \epsilon + c\beta$ is different from Vaudenay's $\pi_\beta(\epsilon) = \epsilon + 2\beta (\bmod\, n+1)$ [Vau], as quoted in [CS03, p. 9].

part of the attack (formally called completeness) of R_4 is shown as Theorem 7.1.13 which follows.

Confidentiality. One needs to show that $k_{pub} = f_R \circ E \circ I(k_{priv})$ is not computationally invertible. For G_4, we have $\delta = I(p, q)$ and the result of the function composition is $e = f_R \circ E(\delta) = \pi_{\beta,c}(\epsilon)$, where

$$\pi_{\beta,c}(\epsilon) = \epsilon + c\beta \quad \text{with} \quad \epsilon = \delta^{-1} \bmod \phi(n).$$

Supposing that p and q are uniformly distributed, $\delta^{-1} \bmod \phi(n)$ is well preshuffled by Assumption 2.2.17′. Instead use Assumption 2.2.18$^{(2)}$, derived from Assumption 2.2.18′: with c and β randomly sampled, it is reasonable to suppose that

$$f_\beta(\delta) = \pi_{\beta,c}\left(\delta^{-1} \bmod \phi(n)\right)$$

is a function family with output computationally indistinguishable from uniform. Furthermore assume that f_β is one-way. Given this, from $f_\beta(\delta)$, it is computationally infeasible to retrieve δ or the pair (ϵ, δ). So it is infeasible to factor n.

Completeness. The following theorem completes intuition provided in the introduction of *Algorithm properties*. It may be interpreted as a generalization of Theorem 6.3.4. Its proof follows closely the related proof of [BDF98, Theorem 3.1], but for β generally not being a power of 2 and a better complexity.

Theorem 7.1.13 (Completeness) *With appropriate choice of β, Steps 4 to 6 of Algorithm R_4 are likely to be successful and run in expected polynomial time in k. An appropriate choice of β is to pick it to be an odd random number of known factorization [Kal03], of size $k/2$, as in the proof of Proposition 7.1.14.*

Proof: Consider $\epsilon\delta - \kappa\phi(n) = \epsilon\delta - \kappa[n - (p+q) + 1] = 1$. Let $g = \gcd(\kappa, \beta)$ and $m = \kappa/g$. Let $\epsilon_0 \equiv \epsilon \bmod \beta$, then $p_0 \equiv p \bmod \beta$ is a solution for x in

$$\begin{aligned} 1 + \kappa[n - x - nx^{-1} + 1] &\equiv \epsilon_0\delta \pmod{\beta}, \qquad (7.12) \\ \kappa x^2 + [\epsilon_0\delta - \kappa(n+1) - 1]x + \kappa n &\equiv 0 \pmod{\beta}, \\ mx^2 - m(p+q)x + mn &\equiv 0 \pmod{\beta/g}. \end{aligned}$$

Because $\gcd(m, \beta/g) = 1$, $m^{-1} \bmod \beta/g$ is well-defined. Then

$$x^2 - (p+q)x + n \equiv 0 \pmod{\beta/g}$$

and by rearranging the terms,

$$\left[x - \frac{p+q}{2}\right]^2 \equiv \left(\frac{p-q}{2}\right)^2 \pmod{\beta/g}. \qquad (7.13)$$

Taking $y = x - (p+q)/2$ in Theorem 7.1.11, a solution for y is a solution for x because the values of

$$a \equiv (p+q)/2 \pmod{\beta/g} \equiv [\epsilon_0\delta - \kappa(n+1) - 1]/(2\kappa)$$

and

$$b \equiv [(p-q)/2]^2 \pmod{\beta/g} \equiv a^2 - n$$

can be computed.

Now consider the algorithm itself. As (e, n) is public, the designer can compute $\epsilon_0 \equiv e \bmod \beta$. The values of κ and δ can be computed via Theorem 7.1.3, with e approximating ϵ, because $|e - \epsilon| \leq \ell_2 n^{3/4-\alpha}$ and $\delta < \ell_1 n^{\alpha}$.

By Theorem 7.1.11, the conditions for Equation 7.12 to be solvable are satisfied by β. Because b is a square, it suffices that $\gcd(b, \beta) = 1$ for b to be a quadratic residue of all odd prime divisors of β. With $b = ((p-q)/2)^2$, either $\gcd(b, \beta/g) = \gcd(p-q, \beta/g) = 1$, or an equivalent solution can be found because if $h = \gcd(b, \beta/g)$ then

$$y^2 \equiv \frac{b^2}{h^2} \bmod \frac{\beta}{gh}.$$

The last three conditions of Theorem 7.1.11 do not apply because $2 \not| \beta$ by hypothesis.

Finalizing the solution of Equation 7.13, with $p_0 \equiv p \bmod \beta$ known, Theorem 7.1.12 establishes that p and q are found in polynomial time $T(k)$. Because the modulo of Equation 7.13 is no larger than Equation 7.12's, the solutions of the former is a superset of the latter's. If x_i is one of the ν solutions to Equation 7.13, then solutions to test for Equation 7.12 are $\{x_i + j\beta/g\}$ for $i \in \{1, ..., \nu\}$ and $j \in \{0, ..., g-1\}$. Note that ν is a small constant [Red96, Chapter 3]. Also $g = \gcd(\kappa, \beta)$, is likely to be small if β is chosen independently of κ. To show this, let β be fixed and suppose that κ is uniformly distributed (although this is a rough approximation). Because $\kappa < \delta < \phi(\beta)$, the probability that κ is relatively prime to β is at least,

$$\frac{\phi(\beta)}{\beta} > \frac{1}{2 \lg \lg \beta} = \frac{1}{2 \lg |\beta|} \gg \frac{1}{|\beta|},$$

which is significantly large. The first inequality comes from the well-known limit:

$$\liminf_{n\to\infty} \frac{\phi(n) \lg \lg n}{n} = e^{-\gamma} \approx 0.561$$

where γ is the Euler-Mascheroni constant [HW79, Theorem 328]. ■

Symmetry. Algorithm (G_4, R_4) is a symmetric backdoor with secret key β.

Cardinality. The following is the strongest result concerning the number of backdoored keys.

Proposition 7.1.14 (Achievable cardinality of KSM II) *Let $\alpha \in \Omega(1)$ and $\alpha < 1/6$. A set of backdoored keys that covers, in total for all runs, $2^{2(1-\alpha)k}$ possible keys is generable. Furthermore, the expected running time can be made of the same order as that of Algorithm G_0 (Proposition 7.1.10 still gives the performance per key).*

Proof: First, the cardinality is analyzed without considering the relevance of a lower bound on α. However, if $\alpha = 0$, there is only one δ, which cannot be secure. In order for the proposition's statement to be correct, the second part of the proof determines this lower bound.

General cardinality. In Step 4 of Algorithm G_4, there are $\ell_1 n^{\alpha}$ choices for ϵ. In Step 6, there are $\frac{2\ell_2}{\beta}\, n^{1-2\alpha}$ choices for c. Therefore there are $2\ell_2\, n^{1-2\alpha}$ choices for c and β together and $2\ell_1\ell_2\, n^{1-\alpha}$ choices for all three parameters (Step 7). This can be interpreted as supposing that for each ϵ, the generated sets of e are distinct and the total number of e is $\Omega\left(2^{2(1-\alpha)k}\right)$.

It remains to subtract the intersections of these sets. Consider a generated $e = \epsilon_1 + c_1\beta_1$ and suppose that another execution of Algorithm G_4 yields the same $e = \epsilon_2 + c_2\beta_2$. Then $\epsilon_1 - \epsilon_2 = c_2\beta_2 - c_1\beta_1$, without the mod $\phi(n)$ operation, because $\epsilon + c\beta < \phi(n)$, with high probability from the choice of sizes of parameters. Count the number of ways in which the following can happen, $\epsilon_1 - \epsilon_2 = c_2\beta_2 - c_1\beta_1$.

To see that the solution for c_1, c_2 is unique [6], consider the extended Euclidean Algorithm on β_1, β_2. Let $g = \gcd(\beta_1, \beta_2)$, then the integers r_1, r_2, such that $r_2\beta_2 - r_1\beta_1 = g$ exist and are unique. Therefore, for fixed ϵ_1, β_1 and ϵ_2, β_2, the solution for c_1 and c_2 is unique, $c_i = r_i(\epsilon_1 - \epsilon_2)/g$. Letting $(\epsilon_1, \beta_1, \epsilon_2, \beta_2)$ vary, there are at most

$$\#\{\delta\}^2 \cdot \#\{\beta\}^2 = 2^{(2\alpha+\frac{1}{2})2k}$$

solutions for them. This can be interpreted as meaning that the size of the intersection between the sets corresponding to ϵ_1 and ϵ_2 is about 2^k, while there are about $2^{2\alpha k}$ such intersections for one set and $2^{2\alpha k}$ such sets.

Finally, subtract the intersection. For an appropriate choice of α, it holds that

$$2^{2(1-\alpha)k} - 2^{(2\alpha+\frac{1}{2})2k} \in \Omega\left(2^{2(1-\alpha)k}\right).$$

For this to hold the first exponent should be strictly larger than the second, that is, such that $1 - \alpha > 2\alpha + \frac{1}{2}$, therefore $\alpha < 1/6$ is required.

Determining the lower bound on α. The smaller α is, the weaker the security of the backdoor becomes. For instance, if $\alpha = 0$ (or close to, depending on the value of ℓ_1), then only one ϵ can be generated (Steps 3 and 4 of G_4). Then, given two keys $e_1 = \epsilon + c_1\beta$ and $e_2 = \epsilon + c_2\beta$, one finds a multiple of β which is $(c_1 - c_2)\beta$.

With an expected number of repetitions (different requests for key regeneration) of $\mathcal{O}(|n/\beta|)$, one finds the value of β. To see this, consider the range of $(c_1 - c_2)$ when

[6] In some cases, the solutions are too large, so they are not collisions, although they are counted as such. Thus the number of collisions is overestimated, for simplicity.

$\alpha = 0$ which is $[-\frac{2\ell_2}{\beta}n, \frac{2\ell_2}{\beta}n]$. Given $(c_1 - c_2)$, the probability that another $(c'_1 - c'_2)$ is relatively prime to $(c_1 - c_2)$ (which would yield β by taking the gcd of both multiples of β) is bounded below by the probability that $(c'_1 - c'_2)$ is a prime number. Since it is randomly chosen, that probability is approximately $1/\ln(n/\beta)$. Therefore the expected number of different $(c'_1 - c'_2)$ generated before finding a suitable one is bounded above by the order of $\ln(n/\beta)$.

What lower bound should be imposed on α? Suppose an arbitrary α, then about $2^{2\alpha k}$ different ϵ can be generated. Let the resulting keys be $e_i = \epsilon_i + c_i\beta$. Via the use of the gcd as above, the probability that the values of two keys allow the extraction of information on β is the probability that $e_i - e_j$ is a multiple of β, which is approximated by:

$$\begin{aligned}\Pr[e_i - e_j = \nu\beta] &\approx \Pr[\epsilon_i + c_i\beta - (\epsilon_j + c_j\beta) = (c_i - c_j)\beta] \\ &\approx \Pr[\epsilon_i = \epsilon_j] \approx \frac{1}{2^{2\alpha k}}\end{aligned}$$

From the point of view of the attacker (the party creating the backdoor), this probability should be negligibly small. Asymptotically, this means that $\alpha \in \Omega(1)$ is required. In practice, to counter exhaustive searches, $2\alpha k > 64$ is required [RSA06, dis02].

Fortunately, it seems, if the distinguisher finds $\nu\beta$, the value of β itself is still unknown. What is a lower bound on the expected number of trials needed to find it? The reasoning is similar to the one above. Given $\nu_1\beta$, what is the probability of finding $\nu_2\beta$ such that $\mu = \gcd(\nu_1, \nu_2) = 1$? (Note, as below, that larger μ's can be considered.) The selection is random and there are $\phi(\nu_1)$ such integers. The value of the Euler function $\phi(\nu_1)$ is bounded above by ν_1 and bounded below by the following well-known property which may be referred in [BS96, p.237, Proposition 8].

$$\phi(\nu_1) \in \Omega\left(\frac{\nu_1}{\log\log \nu_1}\right)$$

The probability of finding $\nu_2\beta$ such that $\gcd(\nu_1, \nu_2) = 1$ is in $\mathcal{O}\left(1/\log\log \nu_1\right)$. The range of ν_1 is $[-\frac{2\ell}{\beta}n^{1-2\alpha}, \frac{2\ell}{\beta}n^{1-2\alpha}]$. As $\alpha < 1/6$ by hypothesis, the expected number of repetitions, $\Omega(|n^{1-2\alpha}/\beta|)$, is not significantly large. Thus, the above requirements,

$$\alpha \in \Omega(1) \text{ or } 2\alpha k > 64, \tag{7.14}$$

are necessary for the security of the backdoor.

Larger μ's can be considered, but this does not affect the above security requirements. Suppose that $\gcd(\nu_1\beta, \nu_2\beta) = \mu\beta$, with $|\beta| \approx 256$ bits. Suppose that exhaustive searches are feasible up to 64 bits [RSA06, dis02]. The problem is to factor $|\mu\beta| \approx 320$ bits ≈ 100 digits, which is about the point where general factorization is difficult [WIF06]. Thus, considering larger μ's does not significantly help the distinguisher. ■

Usually, the β are kept fixed, in determining cardinality. Therefore, denote the key ratio with the β accounted as $\mathcal{R}^*$.

$$2^{-k/3} < \mathcal{R}^*_{G_4} < 2^{-64}.$$

Because there are few restrictions on β, counting them yields about $\#\{\beta\} = 2^{k/2}$ more keys, therefore $2^{-5/6k} < \mathcal{R}_{G_4} < 2^{-64-k/2}$.

Distribution properties. The distribution of e is good, if $\pi_\beta(\delta) = \delta^{-1} + c\beta$ is close to uniform. This is reasonable, from a slightly strengthened Assumption 2.2.26. As in Chapter 2, denote by Assumption 2.2.26′ the combination of Assumption 2.2.26 with a more complete domain for δ, along with the assumption that the addition of $c\beta$ at Step 7 makes the $(2 - 4\alpha)k$ lower bits computationally indistinguishable from uniform, as c is uniformly distributed.

Entropy of embedding. An argument similar to the one for the preceding four algorithms holds. This is gathered by Assumption 2.2.26′, as in *Distribution properties.*

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on δ is stored in e, which by definition depends on such an inverse.

Complexity. Proposition 7.1.9 applies: the complexity is proportional to t_e. Furthermore, no use of a non-instantiated π_β is made. Overall:

$$\begin{aligned} T(G_1) &= t_n + t_e \\ &\approx T(G_0). \end{aligned}$$

Memory. No use of additional memory is made.

Computational assumptions. Assumption 2.2.26 is used in *Distribution properties* and *Entropy.*

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			yes: one-way $\pi_\beta(\epsilon) = \epsilon + c\beta$, Assumption 2.2.18$^{(2)}$	good
completeness			yes: Theorem 7.1.13	good
indisting.	asymmetry		no: β is secret	
	diversity	$\mathcal{R}_{G_4}$	$2^{-5/6k} < \mathcal{R}_{G_4} < 2^{-64-k/2}$ $2^{-k/3} < \mathcal{R}^*_{G_4} < 2^{-64}$	good
		$\mathcal{C}_{G_4}$	$0,$ given Assumption 2.2.26$'$	good
		$\mathcal{V}_{G_4}$	1	good
		$\mathcal{H}_{G_4}$	$1,$ given Assumption 2.2.26$'$	good
	GKR		yes	good
	side-channels	complexity	$T(G_0)$	good
		memory	no	good
computational assumptions			Assumption 2.2.18$^{(2)}$, Assumption 2.2.26$'$	good
simplicity			LS	good

Table 7–7: Properties of Algorithm G_4 of Figure 7–9.

Discussion. The fact that β is generally not a power of 2 is significant. It matters not whether the information transmitted via the backdoor consists of actual bits, or if it is *bits of information.* In R_3, Step 2 performs a modulo operation of a power of 2, which recovers the least significant bits of ϵ. In contrast, R_4 recovers $p_0 \equiv p \pmod{\beta}$ in Steps 4 and 5, for a more general β. The same holds in the comparison of Theorems 6.1.8 and 7.1.12.

For each β of G_3 or G_4, the number of proposed backdoored keys generated is roughly $n^{3/4}$. For G_4, Proposition 7.1.14 improves upon Proposition 7.1.10 because it states that the different values of β produce distinct sets of backdoored keys with a high probability. (No such claim is made for G_1 and G_3 because they are dependent on a choice of unspecified permutation π_β.) G_4 also improves on G_3 because, with comparable key diversity, $\pi_\beta(\delta) = \delta^{-1} + c\beta$ is well-defined.

Relation with Subsection 2.2.8. In Algorithm G_4, the assumed pseudo-random addition of $c\beta$ acts on the $|c\beta| = (2 - 4\alpha)k$ least significant bits of ϵ. Therefore, in Assumption 2.2.26, it is claimed that its $(2 - 4\alpha)k$ most significant bits are computationally indistinguishable from being uniformly distributed.

This means that $x = 2\alpha$ in Theorem 2.2.11. For the theorem to apply so that these $(2 - 4\alpha)k$ most significant bits are properly distributed, it is required that

$$2^{8\alpha k} \leq \delta \leq 2^{(1-4\alpha)2k} \tag{7.15}$$

where δ replaces e in the theorem. However, from Algorithm G_4, $\delta < 2^{2\alpha k}$. This is excluded by the LHS of Inequation 7.15.

Consequently, the conditions of Theorem 2.2.11 are not sufficient to justify the intuition behind Assumption 2.2.26. In fact, a different, perhaps related, theorem would be needed to provide the intuition that Theorem 2.2.11 appeared to.

7.2 Discussion on the first improvement

This section is dedicated to the discussion of the important aspects of the improvements in the previous section. It is presented separately as to highlight these improvements.

7.2.1 Structure of the improvement

Various theorems were useful for our results. The structure of their use is illustrated by Figure 7–10.

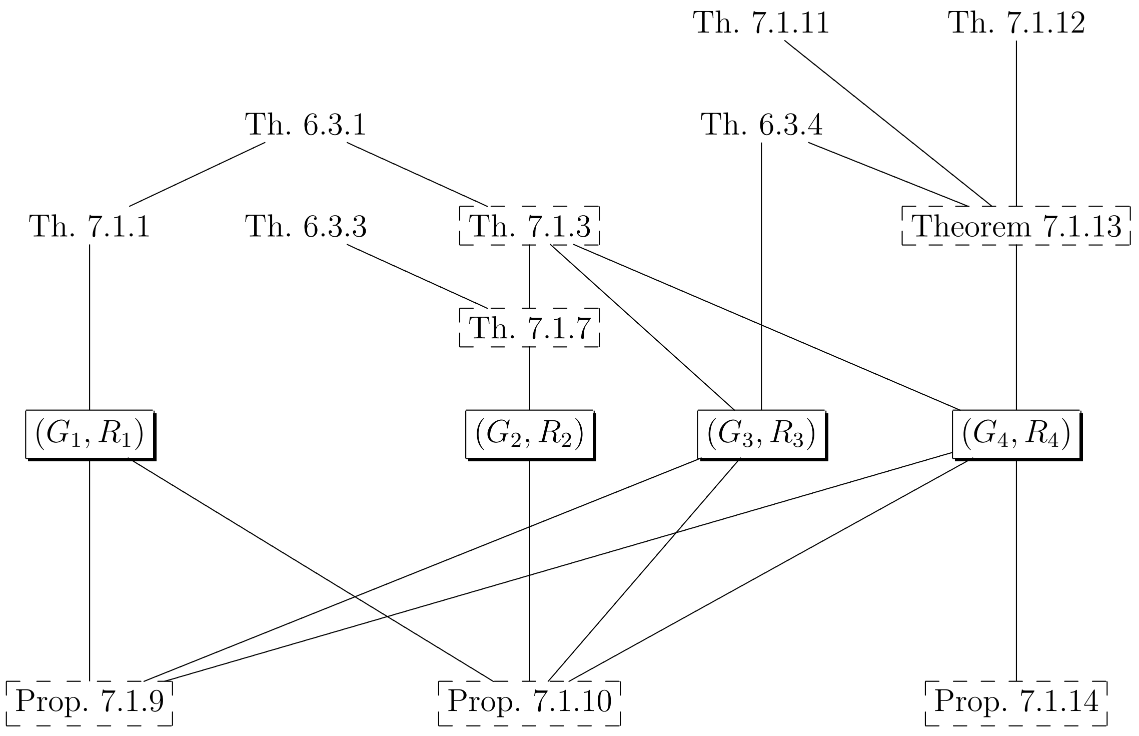


Figure 7–10: Illustration of the theorems' use. The boxes represent our algorithms. The dashed frames highlight our contributions. The three bottom propositions are backdoor properties.

Some improved useful theorems have been presented. Theorems 7.1.3 and 7.1.7 are new improvements of Wiener's Theorem. Theorem 7.1.3 is an extension of Wiener's Theorem that is complementary to Theorem 7.1.1 [BM04], which is concerned with multiplicative perturbations, while ours takes care of additive ones. In comparison, the related generalization of [BM04] yields more information than our extension. Our version of the attack yields the parameters δ and κ, while theirs yields the factorization of n, thus breaking the entire private key. We used this breaking directly for the algorithm (G_1, R_1). For our three remaining algorithms, we achieve the equivalent breaking in different ways. In the algorithm (G_2, R_2), it is through our Theorem 7.1.7, which has the disadvantage of requiring δ to be prime. For the algorithms (G_3, R_3) and (G_4, R_4), it is through the additional use of Theorem 6.3.4 and Theorem 7.1.12 respectively. Every algorithm leaves the bulk of the work to its retrieval algorithm, which does not need to run discreetly.

Theorem 7.1.13 is a generalization of Theorem 6.3.4. Concerning BDF's original Theorem 6.3.4, recall that its first interpretation was that it was unfortunate that the result required non-consecutive bits of ϵ. From its use in the second pair of algorithms, it is fortunate in our context. It is the propagated information that matters and not where it lies.

7.2.2 Failure of simple algorithms

As provided in Section 6.1, there are several simpler proposals which make our algorithms look intricate. In Subsection 6.1.2 for instance, Howgrave-Graham's algorithms embed $p\rceil^{k/2}$ in the public e. Can these backdoored key generators yield a simpler proof for Proposition 7.1.10, that is, a cardinality of at least $2^{\frac{3}{2}k}$?

Generalized key regeneration and variable correlation. Using Theorem 6.1.8 with given $p\rceil^{k/2}$, one finds p. This leaves $\frac{3}{2}k$ bits free, but this approach excludes any capability of generalized key regeneration (GKR) and proper variable correlation, because the bits corresponding to the prime p may remain constant for a fixed β. Our modification with a careful application of π_β remedies to this in a simple way. This is described in *Computational assumptions* of Subsection 6.1.2. A portion of the $\frac{3}{2}k$ free bits is used as a random seed in the pseudo-random function applied on $p\rceil^{k/2}$. Instead of

$$\pi_\beta(p\rceil^{k/2}) : \text{random}, \tag{7.16}$$

the embedded information is

$$\pi_\beta(p\rceil^{k/2} : \text{random}). \tag{7.17}$$

The modification allows an approximate GKR to be performed. By approximate, we mean that it is possible to regenerate values of any individual variable, but we ignore any consequences on other subsequently generated variables. New p or q are regenerated as usual. New e are generated by picking a different *random* parameter in Equation 7.17. This is possible in Equation 7.16, but a fixed p yields fixed upper bits, $\pi_\beta(p\rceil^{k/2})$. Using Equation 7.17 thus allows to regenerate any value of e, with the normal freedom in all of its bits.

However, there still remains two problems. Firstly, even if we were to consider this approximate GKR as sufficient for practical purposes, this modification merely moves the problem of no GKR to that of the security of the π_β function. The distribution of e is dependent on the pseudo-randomness of π_β with key *random*.

Secondly, these values do not satisfy the desirable properties of variable correlation, because of the tie between e and p. For instance, if e is kept fixed, p cannot be regenerated unless one can find

$$e = \pi_\beta(p_1\rceil^{k/2}\!:\!\mathrm{random}_1) = \pi_\beta(p_2\rceil^{k/2}\!:\!\mathrm{random}_2)$$

that is, one needs to find p_2, random_2 such that $\pi_\beta(p_1\!:\!\mathrm{random}_1) = \pi_\beta(p_2\!:\!\mathrm{random}_2)$. This constitutes a collision in a pseudo-random function, π_β, which indicates a weakness in its security. Moreover, even if it was computationally feasible to find a collision in π_β, the distribution of the "weak" p would likely be non-uniform.

To sum up, the security of π_β, needed for approximate GKR, and variable correlation are mutually exclusive. Therefore, if GKR is permitted, Howgrave-Graham's algorithms are very simply distinguished from G_0 as generating backdoored keys.

Simplification of the proof for Proposition 7.1.10. For such an algorithm, consider the required properties of the pseudo-random function, π_β. The cardinality that it produces is computed over its random seeds, the appropriate parameters from the algorithm. The ideal pseudo-random function would be such that on $\frac{1}{2}k$ fixed bits, $\frac{1}{2}k$ random bits and $\frac{1}{2}k$ bits of key β, the image of π_β covers all 2^k possibilities. While this is reasonable, this complicates the choice of π_β.

For simple backdoored key generators, strong results such as Proposition 7.1.10 seem to be achievable only via strong assumptions. For instance, π_β is required to possess the previously mentioned characteristics.

In contrast, our proof of Proposition 7.1.10, only requires assumptions on RSA. Our generation uses inversions modulo the Euler totient function, thus introducing in e a type of “pre-pseudo-randomness” that relates to the pseudo-randomness that is used to conjecture the security of RSA. More precisely, G_4 uses the assumption that for a random $\delta \in \mathbb{Z}^*_{\phi(n)}$ such that $\delta < \ell_1 n^\alpha$,

$$\epsilon \equiv \delta^{-1} \bmod \phi(n) \tag{7.18}$$

is well pre-shuffled (Assumption 2.2.17′). Inversions modulo such $\phi(n)$ are the same as in the definition of RSA (Subsection 1.2.1), which is a candidate trapdoor one-way function and a candidate pseudo-random-like function (which outputs are computationally indistinguishable from uniform), under certain conditions (Proposition 2.2.27).

In order to obtain e, our algorithms apply further operations to these inverses and we conjecture that these perturbations are coherent with this pseudo-randomness (or rather, closeness to the uniform distribution). More precisely, the ϵ of Equation 7.18 becomes

$$e = \epsilon + c\beta.$$

Assumption 2.2.26′ states that such built e are distributed close to uniformly. Assumption $2.2.18^{(2)}$ states that the function doing so is one-way.

7.2.3 Advantages of a reasonable simplicity

The Young-Yung algorithms [YY96, YY97a, YY05a, YY05b], have a higher degree of structure than ours (so far, the ones of Section 7.1). Obviously, less structure is an advantage because the simple algorithms take less coding to write. Simpler also means more flexible and less need for computational assumptions, thus π_β can be very simple. By opposition, Young and Yung [YY96, YY97a, YY05a, YY05b] use strong cryptographic primitives.

Because Assumption 2.2.17′ is reasonable, our algorithms can use a π_β that contributes a pseudo-randomness that is weaker than the other algorithms' cryptographic assumptions. In other words, because RSA-type modular inversion can be assumed to produce well-distributed outputs. Therefore, in order to finally produce an output with distribution computationally indistinguishable from uniform, one only needs to apply to the first outputs (the modular inverses) a scrambling function, π_β, that does not interact with modular inversion itself. This is the case of any function that shuffles its input's bits in a non-algebraic way. For instance, the function can consist in shuffling the bits or XORing them with a fixed random string. The latter is the statement of Assumption 2.2.18′. This is an example of a weaker pseudo-randomness assumption than the ones of the Young-Yung algorithms.

Let $\epsilon = \delta^{-1} \bmod \phi(n)$, as usual so far in this chapter. Consider G_3 with XOR, that is, naively

$$\pi^1_\beta(\epsilon) = \epsilon \oplus \beta'$$

where β' is the concatenation of a certain number of bits of β and a final truncation such that $|\beta'| = |\epsilon|$.

Using Theorems 7.1.3, 7.1.7, 6.3.4 and 7.1.12, one finds the maximum information that is redundant in order to break the key (knowing β). This redundant information corresponds to the parts of of $\pi^1_\beta(\epsilon)$ that can be XORed with random bits. There are many ways of "losing" information, and the cited theorems are not known to list these ways exhaustively. With the same definition of β', let $\beta' \oplus R$ be such that likewise selected bits of β' are "lost". This selection corresponds to the bits of R that are non-zero, as well as randomly picked. So, contrary to β', R is unknown to the designer and serves to "lose" information which can be retrieved via theorems. Then let

$$\pi_\beta^{\oplus}(\epsilon) = (\epsilon \oplus \beta') \oplus R.$$

In other words, R is used to increase entropy, randomizing certain parts of the exponent. The parameter β' introduces non-reversible perturbations (for the distinguisher) and R, reversible ones (for any party).

Similarly, G_3 can be modified with shuffling the bits with a naive

$$\pi_\beta^1(\epsilon) = \sigma_{\beta'}(\epsilon)$$

where σ takes the i^{th} bit ϵ_i and puts it in the $i \oplus \beta'^{th}$ bit position, for $|\beta'| = |\lg \epsilon|$. Finally, with a similar R, a shuffling permutation is

$$\pi_\beta^\sigma(\epsilon) = \sigma_{\beta'}(\epsilon) \oplus R$$

for R selected as precedingly, for π_β^1.

The permutation

$$\pi_\beta(\epsilon) = \epsilon + c\beta$$

of Algorithm G_4 is an example of an instantiation of π_β from G_3. (Recall: Assumption 2.2.26$'$ states that such built e are distributed close to uniform ly. Assumption 2.2.18$^{(2)}$ states that the function doing so is one-way.)

7.2.4 Diversity of the algorithms

The implications of Proposition 7.1.9 are clear; those of Propositions 7.1.10 and 7.1.14 are more subtle, thus call for arguments and further results. A step toward a comparable distribution is getting the cardinality of KSM to be comparable to that of honest keys, that is, close to n (Subsection 7.1.4). From Proposition 7.1.14, about up to $2^{3k/2}$ backdoored keys are generable for a fixed β and up to 2^{2k-64} if the different values of β are accounted. Other results go only up to $2^{3k/2}$ [CS03, YY05a, YY05b] for a fixed β but none appears to be of a form which is suitable for a cardinality that takes different values of β into account.

Consider the set over which the backdoored keys are defined as well as that of the honest keys. Propositions 7.1.10 and 7.1.14 imply that the size of the backdoored set is more comparable to that of the honest one than previously thought. This implies that their distributions are also closer than previously thought, with some assumptions on the distribution of the backdoored set which we discussed to be reasonable in Subsection 2.2.8. For this, the notion of *diversity* of backdoored keys was introduced in Subsection 7.1.4. It encompasses the number of generable keys and a "key entropy", a step toward comparable distributions.

Algorithms CS-2, CS-3 and CS-4 are generalizations of CS-1 via Theorems that generalize Wiener's. These theorems are more powerful than the original one because they require less information to invert the public exponent. Thus, more bits of the published exponent are free, these bits corresponding to the parameter R in the preceding subsection. Because of this, cardinality is increased by about a factor of 3. The three algorithms, CS-2, CS-3 and CS-4, are different ways of obtaining approximately this same cardinality increase. Schematically, we have

$$\begin{array}{rcl} \text{CS-1} & : & e = \pi(\epsilon) \\ & \downarrow & \\ \text{CS-2, CS-3, CS-4} & : & e = \pi(\text{parts of } \delta : \epsilon). \end{array}$$

Our algorithms have greater diversity than CS-1 because they are noisy copies of the ones of that previous system. Our Algorithm G_1 to G_4 have greater cardinality than the CS-1 because they are noisy copies of CS-1. For this to be true for G_4, temporarily consider $G_{4,\pi}$ instead, a version of this last algorithm that uses $e = \pi_\beta(\epsilon + c\beta)$ instead of $e = \epsilon + c\beta$. The R parameter is introduced by G_1 from a multiplicative perturbation, and by the other algorithms, from an additive perturbation. The keys generated are of the form:

$$\begin{array}{rcl} \text{CS-1} & : & e = \pi(\epsilon) \\ & \downarrow & \\ G_1 & : & e = \pi(c \cdot \epsilon) \\ G_2 & : & e = \pi(a + \epsilon) \\ G_3 & : & e = \pi(c \cdot 2^{k/2} + \epsilon) \\ G_{4,\pi} & : & e = \pi_\beta(c\beta + \epsilon) \end{array}$$

and because of this improvement, cardinality is improved by a factor of about 2 or 3.

From CS-1 to G_3, the generalization consists in an additive perturbation, as just mentioned. From G_3 to $G_{4,\pi}$, the generalization concerns the type of bits which are perturbed. In G_3, it is a portion of the MSB bits, i.e. physical bits. In $G_{4,\pi}$, it is general bits: information bits, not physical bits. Also, no bits are kept unperturbed.

$$\begin{array}{rcl} \text{CS-1} & : & e = \pi(\epsilon) \\ & \downarrow & \\ G_3 & : & e = \pi(c \cdot 2^{k/2} + \epsilon) \\ & \downarrow & \\ G_{4,\pi} & : & e = \pi_\beta(c\beta + \epsilon) \end{array}$$

In this sequence of generalizations, cardinality is first increased by a factor 2, and then by a factor 3/2, for an overall factor of 3.

Moreover, the use of π_β in $G_{4,\pi}$ is redundant, given reasonable assumptions. (Recall: Assumption 2.2.26′ states that such built e are distributed close to uniformly. Assumption 2.2.18$^{(2)}$ states that the function doing so is one-way.) Therefore G_4 can be considered the generalization of G_3.

$$\begin{array}{rcl} G_3 & : & e = \pi(c \cdot 2^{k/2} + \epsilon) \\ & \downarrow & \\ G_4 & : & e = c\beta + \epsilon \end{array}$$

The form of G_4 furthermore allow to compute the cardinality for unfixed β. This increases the cardinality by an overall factor of close to 4. However, this special cardinality cannot be compared directly with the other algorithms', as G_4 is the only one for which it is readily computable.

Overall, our algorithms increase the number of generable keys (Subsection 7.1.4), while not allowing any loss of indistinguishability. The cardinality is increased from CS-1 to G_4, while the quality of the other parameters that make up diversity is maintained.

7.3 Second improvement: generalized key regeneration for EG keys

7.3.1 Principles

Generalized key regeneration (GKR) allows the distinguisher to test not only the distribution of each of the parameters, but also their mutual dependencies. Therefore, if a key is composed of parameters that are usually distributed independently, but are not in a backdoored version, then it may be easy for the distinguisher to tell them apart. The analysis of parameter correlations was defined in Subsection 5.1.4.

GKR with EG keys is more complicated than with RSA keys. The former, (p, α, a, β), have more components are there are more relations between them. This is immediate from the honest key generator, given in Figure 7–11.

```
Algorithm $G_0$ [EG key generation]
1: Pick a random prime $p$ of appropriate size.
2: repeat
3:     Pick a random $\alpha \in \mathbb{Z}_p^*$.
4: until $\forall$ prime $p_i$, s.t. $p_i | p-1$, it holds that $[\alpha^{(p-1)/p_i} \bmod p \neq 1]$.
5: Pick a random $0 \leq a \leq p-2$.
6: Compute $\beta = \alpha^a \bmod p$.
7: return $(p, \alpha, a, \beta)$.
```

Figure 7–11: Honest key generation for the EG cryptosystem.

Because of these more complicated relations, the use of *volatile memory* in backdoored key generators appears to be necessary in order to insure GKR.

7.3.2 Improvements

Up to [CS03], GKR had not be taken into account. Therefore, it is only natural that early backdoors as well as the papers of Young and Yung (Section 6.2) do not consider this aspect, or fail this criterion. One of our results accounts for GKR in EG backdoored keys. The corresponding honest key generator is given in Appendix A.2.

Denote a generic encryption function, for instance AES_K or DES_K, as

$$Enc_K : \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_2}.$$

It is used, via

$$f_K : \{0,1\}^{\ell_1} \to \{0,1\}^{|p|},$$

as a pseudo-random function:

$$f_K(x) = [Enc_K(x) : Enc_K^{(2)}(x) : ... : Enc_K^{(|p|/\ell_2)}(x)] \bmod p$$

where the last block, $Enc_K^{(|p|/\ell_2)}(x)$, is truncated in order to obtain a total of exactly $|p|$ bits of output and the trimmed bits are ignored. In other words, the number of encryption blocks is a fraction. This truncation is needed because taking a modulo is many-to-one: without truncation, a portion of the small values of the output would be twice as likely as other values. Therefore, ignoring the outputs above $p - 1$, the mod$p$ is redundant and

$$f_K(x) = [Enc_K(x) : Enc_K^{(2)}(x) : ... : Enc_K^{(|p|/\ell_2)}(x)].$$

Also, at Step 8 of ACK-1, it is understood that the outputs above $p-2$ are similarly ignored.

PARAMETERS:
- Random secret key K.
- Efficiently invertible one-way, pseudo-random function $f_K(x)$.

Algorithm ACK-1 [EG key gen]

```
1: Pick random prime $p$ of appropriate size.
2: Pick a random string $r$ of appropriate size.
3: repeat
4:     Pick a random string $s$ of size $|r|$.
5:     Set $\alpha = f_K(s:r)$.
6: until $\alpha \in \mathbb{Z}_p^*$ and $\forall$ prime $p_i$, s.t. $p_i|(p-1)$,
          $[\alpha^{(p-1)/p_i} \bmod p \neq 1]$.
7: Set $m = 00...0:r$.
8: Set $a = f_K(m)$ and such that $0 \leq a \leq p-2$.
9: Compute $\beta = \alpha^a \bmod p$.
10: return $(p, \alpha, a, \beta)$.
```

Algo. ACK-A-1 [Key retr.]

```
1: Given $(\alpha, \beta, p)$ , compute $(s : r) = f_K^{-1}(\alpha) \bmod p$.
2: Set $m = 00...0:r$.
3: Compute $a = f_K(m)$.
4: while $\beta \neq \alpha^a \bmod p$
5:     Set $m = a\rceil^{|r|+|s|}$.
6:     Set $a = f_K(m)$.
7: return $a$.
```

Figure 7–12: Our symmetric backdoor for EG which embeds a in α.

Algorithm properties.

Confidentiality. This properties is based on the one-wayness of $f_K(x)$. If this holds, then the values of α and a (Steps 5 and 8 of generation, respectively) do not allow the efficient computation of $r:s$.

Completeness. This properties rests on the invertibility of $f_K(x)$. The backdoor information is stored in α and restored at Step 1 of the retrieval algorithm.

Symmetry. The algorithm is a symmetric backdoor with secret key K.

Cardinality. Even assuming the pseudo-randomness of $f_K(x)$, the parameters α and a are not random within a set of the same size as for the honest algorithm. Instead, they are randomly distributed in a smaller set, which is the image of f_K.

We proceed to compare more precisely the cardinalities of the standard ElGamal key generation, G_0, and Algorithm ACK-1. The cardinality of G_0 is

$$\begin{aligned} \#\{(p, \alpha, \beta)\} &= \#\{(p, \alpha, a)\} \\ &= \#\{p\} \cdot \phi(p-1) \cdot (p-1), \end{aligned}$$

where $\#\{\alpha\} = \phi(p-1)$ and $\#\{a\} = p-1$.

The total expected number of keys generated by Algorithm ACK-1 seems to be:

$$\#\{(p, \alpha, a)\} = \#\{p\} \cdot \#\{\alpha\}$$

because for each α, only one a is generable, as they both depend on the same random parameter r.

Nevertheless, accounting the keys from GKR, more keys are generable. These GKR-generable keys are to be counted because GKR is allowed to the distinguisher, therefore that party has access to these keys. We will see that more α are generable via Algorithm ACK-1 and more a, via the algorithm which follows. Therefore

$$\#\{(p, \alpha, a)\} = \#\{p\} \cdot \#\{\alpha\} \cdot \#\{a\}.$$

In some cases of GKR, when a new a is requested for fixed p and α, another algorithm is used, as provided in Figure 7–13. More details are provided in the paragraph *Generalized key regeneration and variable correlation* below.

PARAMETERS:
• A counter c in VM for the number of times that this algorithm has been run, i.e. the number of iterations to obtain the previous a.

Algorithm ACK-GKR-1(K) **regeneration of** a]

```
1: if user asks for new $a$, but keeps $\alpha$ then
2:    Set $(s:r) = f_K^{-1}(\alpha)$.
3:    Set $a = f_K(0...0:r)$.
4:    for $c$ iterations
5:       Set $m = a\rceil^{2|r|}$ and $a = f_K(m)$.
6:    Compute $\beta = \alpha^a \bmod p$.
7:    return $(a, \beta)$.
```

Figure 7–13: GKR for ACK-1. The covered case is: new a and fixed p, α.

At first, it seems that Algorithm ACK-GKR-1 does not generate a significant number of new parameters a because the new ones are functions of the previous ones. On the first run of ACK-1,

$$a = f_K(0...0:r)$$

and on the first run of ACK-GKR-1,

$$a = f_K\left([f_K(0...0:r)]\rceil^{2|r|}\right)$$

and so on. Except the first one, the seeds $\{m\}$ for f_K are truncated outputs of f_K. Therefore, as long as there is no cycle that involves the upper bits of the outputs of f_K, there is no repetitions. The existence of such cycles would show a weakness in the pseudo-randomness of f_K. Assuming no cycles, the number of new a that are such generated is the number of different functions applied (on already counted parameters). Therefore, is it the maximum value of the counter in VM, $2^{|c|}$.

For the backdoored key generator, the number of generable α is given by the fraction of generated α that pass the test at Step 4 of Algorithm G_0, times the total number of generable bit strings via the pseudo-random function (assuming that it is an injection, which is already assumed because it is required to be invertible, thus to be a permutation).

$$\begin{aligned}\#\{\alpha\} &= \frac{\phi(p-1)}{p-1}\cdot 2^{|sr|}\\ &= \frac{\phi(p-1)}{p-1}\cdot 2^{2|r|}.\end{aligned}$$

For both honest and dishonest algorithms, the number of p generated is the same. In the ratio of the cardinalities of KSM and KS, the $\#\{p\}$ terms can be canceled so that

$$\begin{aligned}\mathcal{R}_{G_1} &= \frac{\frac{\phi(p-1)}{p-1}\cdot 2^{2|r|}\cdot 2^{|c|}}{\phi(p-1)\cdot(p-1)}\\ &\approx 2^{|c|+|2(|r|-|p|)}.\end{aligned}$$

Example: For $f_K = $ AES, the lengths $|r| = |s| = 64$ are used, for a total block length of 128, which is the size of the input of f_K.

Distribution properties. The distribution of these backdoored keys is comparable to the honest ones because p is random and, just as for the honest case, the other parameters, α and a, are generated pseudo-randomly. Therefore, assuming pseudo-randomness, the backdoored key distribution of Algorithm ACK-1 is indistinguishable from the one of Algorithm G_0.

If there is regeneration, cases are considered separately, because the distinguisher knows in which case it is. For all cases which will be listed in Table 7–8 below, except cases 2 and 7, regeneration does not produce more backdoored keys, because the same random parameters are used, which are already counted in the diversity of the generation.

In cases 2 and 7, when a is regenerated, while α is not, the situation is similar. The new a is generated from the value of the previous one. As in *Cardinality*, assuming that f_K does not cycle, the a are not repeated and are distributed pseudo-randomly.

Entropy of embedding. From the pseudo-randomness of the f_K function, α and a are pseudo-randomly distributed. Therefore $\mathcal{H}_{G_1} = 1$, given f_K.

Generalized key regeneration and variable correlation. Without GKR, the problem of the designer when retrieving the private key is:

$$\text{from } (p, \alpha, \beta), \quad \text{find } a \tag{7.19}$$

and in the case of this algorithm, the parameter to generate a, that is r, is encrypted in α.

GKR complicates this. Both α and a are functions of r. If α or a is kept fixed and the other is regenerated, then the latter is required to preserve its relation with r. Only with this preservation will the problem of Equation 7.19 remain solvable to the designer.

While r can be retrieved from the parameter which remains fixed, because f_K is invertible, there remains one problem. Because a is regenerated by taking its previous value as a seed (ACK-GKR-1), in order to recover this value, volatile memory is required to store the number of times a has been regenerated.

The distinguisher can request several (p, α, a) or keep any one or two parameters. The seven cases are as in Table 7–8.

	user keeps	user asks new	via algorithm of	description
1	$(\ , \ \ , \ )$	(p, α, a)	Fig. 7–12	
2	$(p, \alpha, \ )$	$(\ , \ \ , a)$	Fig. 7–13	use the $\|r\| + \|s\|$ most significant bits of previous a as a seed to generate a
3	$(p, \ \ , a)$	$(\ , \alpha, \ )$	Fig. 7–12, Steps 3 to 6 (with the same r)	produces at most $2^{\|s\|}$ new α, which are required to be generators of $\mathbb{Z}_p^*$
4	$(\ , \alpha, a)$	$(p, \ \ , \ )$	Fig. 7–12, Step 1 (only)	finds p with generator α
5	$(\ , \ \ , a)$	$(p, \alpha, \ )$	Fig. 7–12, Steps 3 to 6 (with the same r)	finds p and α (a more precise algorithm should insure that $a \leq p-2$)
6	$(p, \ \ , \ )$	$(\ , \alpha, a)$	Fig. 7–12, Steps 2 to 10	
7	$(\ , \alpha, \ )$	$(p, \ \ , a)$	Fig. 7–12, Step 1 (only) then Fig. 7–13	finds p such that α is the is generator of $\mathbb{Z}_p^*$, then finds a as in case 2

Table 7–8: The seven cases of GKR with the algorithms of Figures 7–12 and 7–13.

Is VM absolutely needed? For this type of algorithm where α and a are generated via an f_K, it appears to be the case. A legitimate question is whether new as could be generated as new αs are. In other words, this would mean that the counter is stored in the same way as r or s are. This is the following.

$$\begin{aligned} \alpha &= f_K(0...0\!:\!s\!:\!r) \\ a &= f_K(c\!:\!0...0\!:\!r) \end{aligned}$$

This would allow to regenerate a for a fixed α by picking another value for c. However, c precisely cannot be retrieved by inverting f_K on α. In this case, doing so is needed to solve the problem of Equation 7.19. This is because α contains the only available information on a. Putting c in a similarly generated p yields the same problem, when p and α are kept fixed and a is to be regenerated. Therefore, this type of algorithm appears to require VM. Indeed, there seems to be no room for noise, denoted R in Subsection 7.2.3, so any variation in a has to correspond to information that is retrievable via a public variable (p or α).

Non-volatile memory (NM) is not necessary because if the generator is completely reset, all the values are regenerated from scratch, which is coherent with the honest generator. This property was analyzed as Example 5.2.3.

Complexity. The expected running times of the standard ElGamal key generation and Algorithm ACK-1 are of the same order, assuming that the complexity of f_K is relatively negligible.

The running time of Steps [1,6,9,10] is identical to the one of [1,4,6,7] in G_0. Steps 5 and 8 are similar to 3 and 5 in the original algorithm: both generate pseudo-random strings of the same size. Algorithm ACK-1 has more steps (2, 4 and 7), but their running time is negligible w.r.t. the running time of the standard ElGamal key generation.

If we let $Enc_k = \text{AES}_k$ or DES_k then the time it takes to compute Enc_k is constant, so the time for f_K is bounded by $\mathcal{O}(|p|)$. This time is negligible compared to the time it takes to compute the exponentiation on Step 6 of Algorithm ACK-1.

Therefore, $t_p(G_1) = t_p$, $t_\alpha(G_1) = t_\alpha + T(f_K)$ and $t_a(G_1) = t_a + T(f_K)$.

Memory. GKR uses non-volatile (NV) memory. This property was analyzed as Example 5.2.3.

Computational assumptions. The algorithm uses a family of one-way, pseudo-random functions, f_K, which is not explicitly provided.

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			yes: one-way f_K	good
completeness			yes: f_K is invertible	good
indisting.	asymmetry		no: secret key K	
	diversity	$\mathcal{R}_{G_1}$	$2^{\vert c\vert+2(\vert r\vert-\vert p\vert)}$	good
		$\mathcal{C}_{G_1}$	0, pseudo-random f_K	good
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	1, pseudo-random f_K	average
	GKR		yes: Table 7–8	good
	side-channels	complexity	$t_p(G_1) = t_p$ $t_\alpha(G_1) = t_\alpha + T(f_K)$ $t_a(G_1) = t_a + T(f_K)$	good / F-rel.
		memory	VM (only for GKR)	average
computational assumptions			one-way, pseudo-random f_K	average
simplicity			LS	good

Table 7–9: Properties of the Algorithm ACK-1 of Figure 7–12.

7.4 Third improvement: asymmetric algorithms

Up to now, all our algorithms have been symmetric. In this section, we present asymmetric backdoored key generators for RSA and EG. The advantage is the same as for public key cryptography: the public key is published along with the backdoored key generator's code and, given some assumptions, does not provide the distinguisher with any extra computational advantage.

7.4.1 Principles

In Subsection 6.2.1, are shown the first two asymmetric backdoored key generators for RSA, although they are insecure. Nevertheless, some design principles emerge from them.

Securing the backdoor information

In the algorithm shown in Figure 6–4, the backdoor information, p, is embedded in e. Suppose that E is the designer's encryption function.

Suppose that p was embedded in e, just as in Figure 6–4, but as

$$e = E(p\!:\!r)$$

with r picked at random to fit the leftover input size of E. Then E is used with a secure key length, as listed in Section A.1. Otherwise, either E's key length makes the backdoor unconfidential or forces the generation of a lengthier backdoored key. This would be contradictory to Definition 4.2.9. Therefore a such constructed backdoored key generator would not be one in the strict sense of the definition which we have established and argued made the most sense, in Chapter 4.

Minimizing the backdoor information

Moreover, not all of p is useful when stored in e: by Coppersmith's Theorem (Theorem 6.1.8), only its half upper bits are sufficient. Therefore, the backdoor information can be secured and minimized as

$$e = E(p\rceil^{k/2}\!:r)$$

with r picked at random to fit the leftover input size of E.

Location of the backdoor embedding

There are two choices of public parameters for the embedding, if the backdoor is embedded as a whole (which it should be, in order to conform to GKR): n and e. Suppose that p was embedded in e as

$$e = E(p\!:\!r).$$

This establishes a dependence between the values of e and p. This forbids GKR unless memory is used, as in Subsection 7.3.2.

Suppose instead that p was embedded in n, possibly as

$$n = E(p\!:\!r)$$

with r picked at random to fit the leftover input size of E. Then n would need to be regenerated until both p and q divided n, that is, until

$$q = E(p\!:\!r)/p$$

is prime.

Example 7.4.1 (An algorithm with the preceding principles) Consider a modification of the algorithm of Figure 6–4 that uses RSA encryption directly in RSA backdoored key generation. Let the designer's RSA public key be (N, E) and its private key be D, with $|N| = 2k$. The result is secure, but has a relatively low cardinality. We later on improve upon it.

PARAMETERS:
- The designer's RSA public key is (N, E) with private key D and $|N| = 2k$.

```
Key generation algorithm
 1: repeat
 2:     Pick rp $p$ of size $k$.
 3:     Pick random $r$ of size $k$.
 4:     Set $n \equiv (p:r)^E \bmod N$.
 5: until $q = \lfloor n/p \rfloor$ is prime.
 6: repeat
 7:     Pick a random odd $e$ such that $|e| \leq 2k$.
 8: until $\gcd(e, \phi(n)) = 1$.
 9: Compute $d \equiv e^{-1} \bmod \phi(n)$.
10: return $(p, q, d, e)$.
```

```
Key retrieval
1: Input of $(n, e)$.
2: Set $(p:r) \equiv e^D \bmod N$.
3: Compute $d \equiv e^{-1} \bmod \phi(n)$.
4: return $d$.
```

Figure 7–14: Modification of Figure 6–4, from Young and Yung (1996): a backdoor for RSA that encrypts p within n.

Selected algorithm properties. Some properties are interesting to analyze. It is useful to notice that confidentiality holds and that GKR is feasible. On the other hand, cardinality appears low in comparison to our best symmetric algorithms. Nevertheless, it is close to our best cardinality for asymmetric algorithms.

Confidentiality. The key lengths are secure, because the designer's security parameter is the same as the distinguisher's. The input of E has size $|N| = |p| + |r| = 2k$.

Cardinality. The total number of backdoored keys may seem to be as in Example 2.2.9:

$$\approx 2^{4k-2\lg k}$$

that is, the same as the number of honest keys. However, it is not the case: n needs to be regenerated until both p and q divided n, that is, until

$$q = E(p\!:\!r)/p$$

is prime. For an individual r, this probability is small, but it is useable over all r.

Suppose that there is sufficient randomness in r to assumably produce all the random values that correspond to q's size, that is, $|r| = |p|$. This division, $E(p\!:\!r)/p$, succeeds with good probability because there are about 2^k multiples of a k-bit p that have length at most $2k$. However, by Theorem 2.2.4, the expected number of p is $2^{k-\lg(k)}$. So not all values of $2k$-bit integers are covered. The expected number of covered $2k$-bit integers is

$$\#\{p\!:\!r\} = \#\{p\} \cdot \#\{r\} = 2^{k-\lg(k)} \cdot 2^k = 2^{2k-\lg(k)}$$

and these are distributed indistinguishably from as from a uniform distribution, because $E(p\!:\!r)$ is close to uniform by Proposition 2.2.27.

Then p divides $E(p\!:\!r)$ with probability given by the fraction of strings covered by multiples of p times the expected fraction of strings covered by the image of E.

This is:

$$\begin{aligned}
\Pr\left[\frac{E(p\!:\!r)}{p} \in \mathbb{Z}\right] &= \frac{\#\{2k\text{-bit multiples of } p\}}{2^{2k}} \cdot \frac{\#\{p\!:\!r\}}{2^{2k}} \\
&= \frac{2^k}{2^{2k}} \cdot \frac{2^{2k-\lg(k)}}{2^{2k}} = \frac{1}{2^{k+\lg(k)}}. \qquad (7.20)
\end{aligned}$$

By Theorem 2.2.4, a k-bit integer is prime with probability of about $2^{-\lg(k)}$. Therefore

$$\Pr\left[\frac{E(p\!:\!r)}{p} \text{ is prime}\right] = \frac{1}{2^{k+\lg(k)}} \cdot 2^{-\lg(k)} = \frac{1}{2^{k+2\lg(k)}}. \qquad (7.21)$$

Overall, the expected number of backdoored keys per value of e (which is unaffected) is

$$\begin{aligned}
\mathcal{N}_{G_1,n} &= \#\{p\!:\!r \mid \exists \text{ prime } q \text{ s.t. } pq = E(p\!:\!r)\} \\
&= \#\{p\} \cdot \#\{r\} \cdot \Pr\left[\frac{E(p\!:\!r)}{p} \text{ is prime}\right] \\
&= 2^{k-\lg(k)} \cdot 2^k \cdot \frac{1}{2^{k+2\lg(k)}} \\
&= 2^{k-3\lg(k)}.
\end{aligned}$$

In the ratio of the cardinalities of KSM and KS, the $\#\{e\}$ terms can be canceled so that

$$\mathcal{R}_{G_1} = \frac{\mathcal{N}_{G_1,n}}{\mathcal{N}_{G_0,n}} \approx \frac{2^{k-3\lg k}}{2^{2k-2\lg k}} = 2^{-k-\lg k}.$$

Generalized key regeneration and variable correlation. As opposed to the previous suggestion of embedding, $e = E(p\!:\!r)$, GKR is allowed. The only non-trivial case is to keep q and find a new p. To regenerate, one swaps the roles of p and q and regenerates with q as the backdoor value to be embedded. □

What are the more favorable encodings of $(p:r)$ in E? Is it possible to have a shorter $E(p:r)$ so that there remains space for random bits, r', in $n = E(p:r):r'$. If it were possible, then n could correspond to many choices of multiples of p, which is desirable as it is of the form $n = pq$. One way is to truncate $E(p:r)$, but the feasibility of inverting a truncated RSA encryption is, to our knowledge, an open question.

Example 7.4.2 (Leaving space for r', but insecurely) Let

$$n = E(p\rceil^{k/2}):r'$$

for $E(x) = x^3 \bmod N$ and an assumed secure-length RSA modulo N. Then

$$|E(p\rceil^{k/2})| = 3k/2$$

so the $k/2$ MSB are 0. Therefore $|r'| = k/2$ bits of randomness can be appended. However, confidentiality does not hold, because 3 is an easily invertible choice of RSA public exponent, as 3rd roots modulo N are feasibly computable. □

This last example illustrates how it does not appear to be possible to use RSA encryption directly in RSA backdoored key generation. Even if only $|p\rceil^{k/2}| = k/2$ bits are encrypted, the resulting encryption is $2k$ bits long. It seems that the loop from Step 1 to 5 of the algorithm of Figure 7–14 remains needed in order to find an r such that

$$E(p\rceil^{k/2}:r) = \text{multiple of } p.$$

Optimal location of the randomness

Distribution properties are a constant issue when RSA is used to embed a backdoor in RSA keys. This is directly consequent of the location of the embedding and of the generation of the rest of the parameters.

A way to solve this is to use a different cryptosystem to instantiate E. Since the length of the public parameters is $\ell_{p(G_0)} = 2k$ and since security requires that $|N| = \ell_{p(G_0)}$, any randomness is required to be used inside the designer's encryption function, E. Therefore, we seek another $E : \{0,1\}^{n(\ell_E)} \to \{0,1\}^{m(\ell_E)}$. Function E shall satisfy Definition 4.2.5, so its output is close to uniform, and thus uses internal randomness, implicit here. It is also such that $m(\ell_E) \ll \ell_{p(G_0)}$ so that p can be embedded in n, as

$$n \;=\; r_1 : E(p) : r_2 \tag{7.22}$$

with r_1, r_2 picked at random to fit the leftover $\ell_{p(G_0)}$ size of n. This allows the division

$$n = qp + r_o$$

with quotient q and remainder r_o. Then the subtraction $r_2' = r_2 - r_o$ does not affect $E(p)$ and the picked p, if r_2 is sufficiently long. By the Euclidean Algorithm, $|r_o| \le |q| = k$, so $|r_2'| \ge k$ is required. Therefore $|r_2| = k$. The probability that

$$q \;=\; \frac{n - r_o}{p} = \frac{r_1 : E(p) : r_2'}{p} \tag{7.23}$$

is prime is approximately given by Theorem 2.2.4, as much as r_2' approximates r_2.

Example 7.4.3 (Modified algorithm properties) Accordingly, let G'_1 be the same as the algorithm of Figure 7–14 of Example 7.4.1, but with E replaced as described: Step 4 is replaced by an assignment which corresponds to Equation 7.22 and Step 5's computation of q is done accordingly to Equation 7.23.

$$n = r_1 : E(p) : r_2$$

Some properties are improved and the others are the same as for the algorithm of Figure 6–18. The notation ought to be more precise. Because only i bits of p are encrypted, those sufficient to recover it via a given theorem, change $E(p)$ to $E \circ I(p)$. Also, suppose that the random suffix has length $|r_2| = k$, therefore, the random prefix has length $|r_1| = k - i$. Also, r_1 has the distribution of the MSB of the product of two primes.

Modified distribution properties. This property is improved and its analysis is almost the same as for the algorithm of Figure 6–18. The key generation algorithm produces instances of n that are the product of a truly random p and a somewhat random q. Amongst other qualities, the upper $|r_1|$ bits of n have the correct distribution of such a product. Therefore, $\mathcal{C}_{G'_1} \approx 0$, although further knowledge on the precise distributed of n should determine whether this value is truly negligibly small.

Modified cardinality. The expected cardinality is silghtly improved (the following analysis resembles the one that led to Equation 6.1 for PAP). The only part of G_1's analysis which changes for G'_1 is Equation 7.20. Its first term, the fraction of strings covered by multiples of p, is replaced by a value of 1, because substituting r_2

by r_2' allows for the exact division by p.

$$\Pr\left[\frac{r_1 \colon E \circ I(p) \colon r_2'}{p} \in \mathbb{Z}\right] = 1$$

Therefore, Equation 7.21 becomes

$$\begin{aligned}\Pr\left[\frac{r_1 \colon E \circ I(p) \colon r_2'}{p} \text{ is prime}\right] &= 1 \cdot 2^{-\lg k} \\ &= \frac{1}{2^{\lg k}}.\end{aligned}$$

The remaining part of the analysis is

$$\begin{aligned}\mathcal{N}_{G_1',n} &= \#\{r_1 \colon p \colon r_2' \mid \exists \text{ prime } q \text{ s.t. } pq = E(p \colon r)\} \\ &= \#\{r_1\} \cdot \#\{p\} \cdot \#\{r_2\} \cdot \Pr\left[\frac{r_1 \colon E \circ I(p) \colon r_2'}{p} \text{ is prime}\right] \\ &\approx 2^{k-i} \cdot 2^{i-\lg k} \cdot 1 \cdot 2^{-\lg k} \\ &= 2^{k-2\lg k}. \end{aligned} \tag{7.24}$$

In the ratio of the cardinalities of KSM and KS, the $\#\{e\}$ terms can be canceled so that

$$\begin{aligned}\mathcal{R}_{G_1'} &= \frac{\mathcal{N}_{G_1',n}}{\mathcal{N}_{G_0,n}} \approx \frac{2^{k-2\lg k}}{2^{2k-2\lg k}} \\ &= 2^{-k}. \end{aligned} \tag{7.25}$$

□

7.4.2 Improvements

The following algorithm is another contribution. The designer has no use for a secret key β in the following RSA backdoored key generator. In order to use accepted secure length for RSA, AES and ECIES, let $k = 512$.

PARAMETERS:
- E^{ECIES} is the Elliptic Curve Integrated Encryption Scheme's public-key encryption function and D^{ECIES} is the corresponding decryption function.

Algorithm ACK-2 [RSA key gen]

```
 1: Pick random $(K, x) \in \{0,1\}^{128} \times \{0,1\}^{32}$.
 2: Set $p]^{k/2} = AES_K(x) : AES_K^{(2)}(x)$.
 3: repeat
 4:     Pick random $p|_{k/2} \in \{0,1\}^{256}$.
 5: until $p$ is prime.
 6: repeat
 7:     Pick random $r_1 \in \{0,1\}^{192}$, $r_2 \in \{0,1\}^{512}$.
 8:     Set $n = (r_1 : E^{ECIES}(k : x) : r_2)$.
 9: until $q = \lfloor n/p \rfloor$ is prime.
10: repeat
11:     Pick a random odd $e$ such that $e < \phi(n)$.
12: until $\gcd(e, \phi(n)) = 1$.
13: Compute $d \equiv e^{-1} \bmod \phi(n)$.
14: return $(p, q, d, e)$.
```

Algorithm ACK-A-2 [Key retr.]

```
1: Input of $(n, e)$.
2: Set $(K : x) = D^{ECIES}(n]_{513}^{833})$.
3: Set $p]^{k/2} = AES_K(x) : AES_K^{(2)}(x)$.
4: Find $p$ using $p]^{k/2}$ [Theorem 6.1.8].
5: Set $q = \lfloor n/p \rfloor$.
6: return $(p, q)$.
```

Figure 7–15: An asymmetric backdoor for RSA using ECIES and AES. The information on p is stored in n.

Algorithm properties.

Confidentiality. This property is achieved via public-key encryption with ECIES.

Completeness. This property is achieved via a standard encryption function as well as via one of Coppersmith's factorization method from approximate values of p (Theorem 6.1.8).

Symmetry. This backdoored key generator is asymmetric because the backdoor information is encrypted with ECIES and this cryptosystem is asymmetric.

Cardinality. Change the parameter lengths to asymptotic notation, using $2k = 1024$ and assuming that the secure lengths for the other cryptosystems vary proportionally. Therefore, $(K, x) \in \{0,1\}^{k/4} \times \{0,1\}^{k/16}$ so $i = 5k/16$.

Nevertheless, the analysis that leads to Equation 7.25 holds. In the ratio of the cardinalities of KSM and KS, the $\#\{e\}$ terms can be canceled so that $\mathcal{R}_{G_1} = 2^{-k}$.

Distribution properties. This property's analysis is almost the same as for the algorithm analyzed in Example 7.4.3. The key generation algorithm produces instances of n that are the product of a truly random p and a somewhat random q. Amongst other qualities, the upper $|r_1|$ bits of n have the correct distribution of such a product. Therefore, $\mathcal{C}_{G_1} \approx 0$, although further knowledge on the precise distributed of n should determine whether this value is truly negligibly small.

Entropy of embedding. From the pseudo-randomness of the E^{ECIES} function (Proposition A.3.6), the middle bits of n are pseudo-randomly distributed, but the same does not apply to the lower bits, as they must be adjusted so that q is prime. This yields the evaluation: $0 < \mathcal{H}_{G_1} < 1$, given EG.

Generalized key regeneration and variable correlation. This property is easily satisfied because all parameters are generated independently. The backdoor information on p is stored in n, which by definition depends on such an inverse.

Complexity. Following the asymptotic notation of *Cardinality*, this property is as for the preceding algorithm, with RSA-$2k$ replaced by ECIES-$5k/16$: The generation of e is the same as for G_0, so $t_e(G_1) = t_e$. For the generation of n, Step 9 is done efficiently as shown to obtain Equation 7.23.

$$t_p(G_1) \quad = \quad t_p + T(\text{AES}) = t_p$$

and

$$t_n(G_1) \quad = \quad T(\text{ECIES-}5k/16).$$

Memory. No use of additional memory is made.

Computational assumptions. The pseudo-randomness of AES and the encryption scheme ECIES are used.

Simplicity. There is about as much structure as in the preceding algorithm, which is rated LS.

confidentiality			yes: ECIES	good
completeness			yes: f_K and Th. 6.1.8	good
indisting.	asymmetry		yes: ECIES is asymmetric	
	diversity	$\mathcal{R}_{G_1}$	2^{-k}	poor*
		$\mathcal{C}_{G_1}$	0	good
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	$0 < \mathcal{H}_{G_1} < 1$, given EG	poor
	GKR		yes	good
	side channels	complexity	$t_p(G_1) = t_p$ $t_q(G_1) = T(\text{ECIES-}5k/16)$ $t_e(G_1) = t_e$	good
		memory	no	good
computational assumptions			AES and ECIES	average
simplicity			LS	good

Table 7–10: Properties of the Algorithm ACK-2 of Figure 7–15. (*) Note that the ratio is good, for an asymmetric algorithm.

A similar algorithm is given for backdoors in EG key generation. Following the standard accepted key lengths of Section A.1, let $|p| = 997$ (which is the length of the ElGamal private key). The symmetric scheme AES is used with key length 128. Let $f_K(x)$ be as for the preceding EG algorithm (Subsection 7.3.2):

$$f_K(x) = [Enc_K(x) : Enc_K^{(2)}(x) : \ldots : Enc_K^{(8)}(x)]$$

with tail truncation and $Enc_K = AES_K : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$. Therefore, $f_K : \{0,1\}^{128} \rightarrow \{0,1\}^{997}$.

PARAMETERS:

• E^{ECIES} is the Elliptic Curve Integrated Encryption Scheme's public-key encryption function and D^{ECIES} is the corresponding decryption function.

```
Algorithm ACK-3 [EG key gen]
1: Pick **rp** $p$ of appropriate size.
2: **repeat**
3:     Pick random $(K, x) \in \{0,1\}^{128} \times \{0,1\}^{32}$.
3:     Pick random $r \in \{0,1\}^{677}$.
4:     Set $\alpha = E^{ECIES}(k:x):r$.
5: **until** $\alpha \in \mathbb{Z}_p^*$ and $\forall$ prime $p_i$, s.t. $p_i | p-1$
           and $\left[\alpha^{(p-1)/p_i} \bmod p \neq 1\right]$.
6: Set $a = f_K(x)$ and such that $0 \leq a \leq p-2$.
7: Compute $\beta = \alpha^a \bmod p$.
8: **return** $(p, \alpha, a, \beta)$.
```

```
Algorithm ACK-A-3 [Key retr.]
1: Input of $(p, \alpha, \beta)$.
2: Let $(k:x) = D^{ECIES}(\alpha\rfloor_{320})$.
3: Let $a = f_K(x)$.
4: Return $a$.
```

Figure 7–16: An asymmetric backdoor for EG using ECIES and AES. The information on a is stored in α.

Algorithm properties. The encryption of the parameters that determine the private key a are in the 320 first bits of α.

Confidentiality. This property is achieved via public-key encryption with ECIES.

Completeness. This property is achieved via a standard encryption function.

Symmetry. This backdoored key generator is asymmetric because the backdoor information is encrypted with ECIES and the cryptosystem is asymmetric.

Cardinality. Change the parameter lengths to asymptotic notation, using $|p| = 997$ and assuming that the secure lengths for the other cryptosystems vary proportionally. Therefore, $(K, x) \in \{0,1\}^{160} \approx \{0,1\}^{|p|/6}$ and $r \in \{0,1\}^{677} \approx \{0,1\}^{2|p|/3}$. In asymptotic notation, this means that there are $\#\{\alpha\} \approx 2^{5|p|/6}$. Because the (K, x) are reused to generate a, there is one such parameter per parameter α.

In the ratio of the cardinalities of KSM and KS, the $\#\{p\}$ terms can be canceled so that $\mathcal{R}_{G_1} = 2^{-7|p|/6}$.

Distribution properties. By Proposition A.3.6, α is correctly distributed. Assuming the pseudo-randomness of AES, a is correctly distributed.

Entropy of embedding. From the pseudo-randomness of the EG encryption (Proposition A.3.6) and the one of f_K, $\mathcal{H}_{G_1} = 1$, given EG and AES.

Generalized key regeneration and variable correlation. The regeneration cases are similar to the ones of Algorithm ACK-1 (Figure 7–12). Volatile memory is required for some cases, for storing the string (K, x), which is comparable to the parameter r of Algorithm ACK-1.

Complexity. Following the asymptotic notation of *Cardinality*:

$$t_\alpha(G_1) \quad = \quad t_\alpha \cdot T(\text{ECIES-}|p|/6)$$

and

$$t_a(G_1) \quad = \quad T(f_K).$$

Memory. GKR uses non-volatile (NV) memory.

Computational assumptions. The algorithm uses assumptions on AES and ECIES.

Simplicity. The algorithm has a relatively low structure. This is intuitively apparent, even without comparing with other algorithms, because it consists of almost the same steps as the honest algorithm.

confidentiality			yes: ECIES	good
completeness			yes: f_K and Th. 6.1.8	good
indisting.	asymmetry		yes: ECIES is asymmetric	
	diversity	$\mathcal{R}_{G_1}$	$2^{-7\vert p\vert/6}$	average
		$\mathcal{C}_{G_1}$	0, given ECIES and AES	good
		$\mathcal{V}_{G_1}$	1	good
		$\mathcal{H}_{G_1}$	1, given ECIES and AES	good
	GKR		yes	good
	side channels	complexity	$t_p(G_1) = t_p$ $t_\alpha(G_1) = t_\alpha \cdot T(\text{ECIES-}\vert p\vert/6)$ $t_a(G_1) = T(f_K)$	poor
		memory	VM (only for GKR)	average
computational assumptions			AES and ECIES	average
simplicity			LS	good

Table 7–11: Properties of the Algorithm ACK-3 of Figure 7–16.

Discussion. The expected running time of this algorithm can be improved by repeating Step 4 when Step 5 fails its loop exiting condition. The repetitions should be limited to a constant number of times, in case the failure is due to the other previously picked parameters. Also, in f_K, one could use $Enc_K(x) = E^{ECIES}(K:x)$ to reduce the number of security assumptions, at the cost of a worst complexity.

It would have been possible to encrypt $(K:x)$ into p, but many key generation requirements include more restrictions on p than the basic ones, so that there are more possible α than p. Moreover, if primes p with known factorization of $p-1$ are generated, using a result from Kalai [Kal03], then Step 5 can be sped up. As for the other public parameter, β, it is perhaps unusable for this purpose since it is computed from α and a (the discrete logarithm is involved).

7.5 Classification of backdoors

For complexity, some algorithms have two ratings. For example, the analysis of And-93 is such that assuming that $T(F) \in \mathcal{O}(t_{\sqrt{n}})$, the complexity is rated poor. Without this assumption, the complexity is rated as F-relative. Thus overall, this situation is denoted as "poor / F-relative". In general, the first component of this notation is the result of the complexity analysis, assuming that the function F has relatively negligible complexity.

Some measures are compilations of more detailed ones. A criterion which has many sub-criteria is rated as its lowest-ranked sub-criterion. For instance, the diversity analysis of PP is such that the complexity is rated "good / F-relative" and the memory is rated "average", therefore the cumulative low is "average / F-relative". Similarly for side channels, the rating is the worst of the complexity and the memory.

The diversity is rated as the worst of the key ratio and the variable correlation. If the distribution properties or the entropy fails, then an exponent of (-) is appended to any non-failing rating, because they are more difficult to test. This is coherent with our suggested definition of diversity, given as Definition 5.1.15.

For asymmetric algorithms, an exponent of $(*)$ is appended to the name.

7.5.1 Backdoors to RSA key generation

Compare backdoored key generators for RSA.

Algorithm		confi-dent.	diver-sity	GKR	side channels	assump-tions	simplicity
Name	Table						
And-93	6–1	failed	poor	good	poor / F-relative	average	good
HG	6–2	good	failed	failed	good / F-relative	average	good
YY96-RSA$^{(*)}$	6–3	failed	failed	failed	poor	good	good
PAP$^{(*)}$	6–4	failed	poor	failed	good / F-relative	average	average
PAP-2$^{(*)}$	6–7	failed	poor	good	good / F-relative	poor	average
PP	6–8	failed	poor	good	average / F-relative	poor	average
EC-SETUP$^{(*)}$	6–9	good	poor	good	failed	poor	failed
CS-1	6–10	good	failed	good	failed	average	good
CS-2	6–11	good	poor	good	good / F-relative	average	average
CS-3	6–12	good	poor	good	good / F-relative	average	average
CS-4	6–13	good	poor	good	poor / F-relative	average	good
CS-5	6–14	good	failed	good	failed	average	average
G_1	7–3	good	good	good	good / F-relative	average	good
G_1'	7–4	good	good	good	good / F-relative	average	good
G_2	7–5	good	good	good	good / F-relative	average	good
G_3	7–6	good	poor	good	good / F-relative	average	good
G_4	7–7	**good**	**good**	**good**	**good**	**good**	**good**
ACK-2$^{(*)}$	7–10	**good**	**poor**	**good**	**good**	**average**	**good**

Table 7–12: Comparison of backdoored key generators for RSA.

7.5.2 Backdoors to EG key generation

Compare backdoored key generators for EG.

Algorithm		confidentiality	diversity	GKR	side channels	assumptions	simplicity
Name	Table						
YY96-EG$^{(*)}$	6–5	failed	failed	failed	poor	average	good
YY96-EG-pure	6–6	good	failed	failed	failed	average	average
ACK-1	7–9	**good**	**good**	**good**	**average** / F-relative	**average**	**good**
ACK-3$^{(*)}$	7–11	**good**	**average**	**good**	**poor**	**average**	**good**

Table 7–13: Comparison of backdoored key generators for EG.

7.5.3 Location of the embedding

The proposals of Crépeau and Slakmon [CS03], more related to our work, can be put in two categories. First, the proposals in which the information necessary for the factorization of an RSA modulus, n, is hidden in the public exponent, e are *hidden exponent methods.* Second, the proposals in which the factorization is hidden in n itself are *hidden modulus methods.* Both approaches have their own merits. The latter, which includes the (direct) hidden factorization methods has the advantage of a running time proportional to the original algorithm, but is still vulnerable to timing attacks as no known speedup for the generation of n applies.

One advantage to a hidden modulus method is to allow the use of a common industry-standard choice for e, that is, $e = 65537 = 2^{16} + 1$, which has low Hamming weight and facilitates fast exponentiation. However, this concerns a particular applied case, not the general theoretical cryptosystem, as defined in [PKC03]. In the light of such particular applied cases, the choice of letting e vary may be interpreted as a tradeoff. In our algorithms, achieving a backdoor that resists timing analysis comes at the cost of increasing the time for all parties to encrypt data to a particular party. This may be useful for backdoors used for administration purposes, such as password recovery. Also, using a fixed e is generally not secure [7] (As already pointed out in Subsection 2.2.8).

The other category of methods, comprising the hidden exponent methods, has the advantage of allowing an arbitrary generation of n and of permitting speedups in the generation of the primes [JPV00]. The present results, in the form of Algorithms G4, ACK-2 and ACK-4, are the first to have a running time strictly proportional to their respective original key generation algorithm. Thus with our results, timing analysis is a less useful tool and the distinguisher, leading this "backdoor detection attack" on its own keys, has a lesser chance of distinguishing backdoored keys.

7.5.4 Symmetry and asymmetry

We presented the first asymmetric algorithms for RSA and EG. For RSA, other algorithms, had been presented as such, but failed important security criteria. Moreover, the algorithm of [YY05a] is not fully asymmetric: Algorithm PP relies on a

[7] For instance, given e different encryptions of a message m, it is trivial to recover the message [Sti06, Exercise 5.17].

private value i. The use of i may not be necessary, but this algorithm fails the critical criterion of confidentiality.

7.6 Chapter notes

Section 7.1. First improvement: complexity and diversity. The results in Section 7.1 have been submitted for publication as [ACS].

Theorems 7.1.3 and 7.1.7 are contributed by Crépeau and Slakmon, even though Subsection 7.1.2 was compiled in [ACS]. The proof of Theorem 7.1.12 is taken from [BDF98]. It is a slight generalisation of a result of [Cop96].

Section 7.3. Second improvement: generalized key regeneration. The symmetric EG backdoor of Subsection 7.3.2 is the work of R. Kazmi. Its analysis and the one of its GKR is joint work with the author of this thesis.

Section 7.4. Third improvement: asymmetric algorithms. The results in Section 7.4 have been submitted for publication as [ACK].

CHAPTER 8
Conclusion

The aim of this work was to push further the mathematical knowledge about the security of RSA. We have shown a positive result that improves on the security of padding schemes for RSA signatures. This consists in Theorem 3.3.1 and its applications.

We have also shown a negative result that implies that more RSA keys are weak, in the sense that they are subject to having being contaminated by an illegitimate backdoor. In the process, we have given or shown the following improvements on definitions pertaining to backdoors. The definitions from the kleptography papers are summarized in Table 4–2.

	improvement
general definition	Definition 4.2.9 is more complete than any kleptography definition (Table 4–2)
indistinguishability of keys	Definition 5.1.15 is more complete than Definition 5.1.1
side channels	Section 5.2 is more complete than the analysis of [CS03]
simplicity and computational assumptions	Section 5.3 presents a new analysis
general comparision of properties	Table 5–5 is a wieldy way of summarizing properties, as opposed to enumerating them

Table 8–1: Summary of this work's definitions on backdoors.

The following are improvements on algorithms pertaining to backdoors. The algorithms from the kleptography papers are summarized in Table 4–3

	improvement
pseudo-code	Chapters 6 and 7 present algorithms in uniformed pseudo-code
complexity, diversity and computational assumptions	for RSA, the algorithms of Section 7.1 improve on CS-1 and CS-3 of Subsection 6.3
generalized key regeneration (GKR)	for RSA, the algorithms of Section 7.1 improve on the ones of Subsection 6.1.2
	for EG, the algorithms of Section 7.3 improve on the ones of Subsection 6.2.1
asymmetry	the algorithms of Subsection 7.4 improve on all other algorithms, for both RSA and EG
comparison	Table 7–12 is a comparison of backdoored key generators for RSA.
	Table 7–13 is a comparison of backdoored key generators for EG.

Table 8–2: Summary of this work's main algorithmic improvements for backdoors.

The following are improvements on theorems and propositions pertaining to backdoors.

	improvement
Proposition 7.1.9	for RSA, the complexity is **linear** in the one of G_0 and the one of π_β, a keyed pseudo-random function: $T(G_1) \approx T(G_0) + T(\pi_\beta)$
Proposition 7.1.10	for RSA, with the complexity of Proposition 7.1.9, cardinality is as high as any other algorithm (except CS-4 is slightly higher)
Proposition 7.1.14	for RSA, with the complexity of Proposition 7.1.9, **cardinality is the highest** known
	Theorem 7.1.13 shows the completeness of an algorithm achieving Proposition 7.1.14 (and this theorem may be interpreted as a generalization of Theorem 6.3.4)

Table 8–3: Summary of this work's theorems and propositions on backdoors.

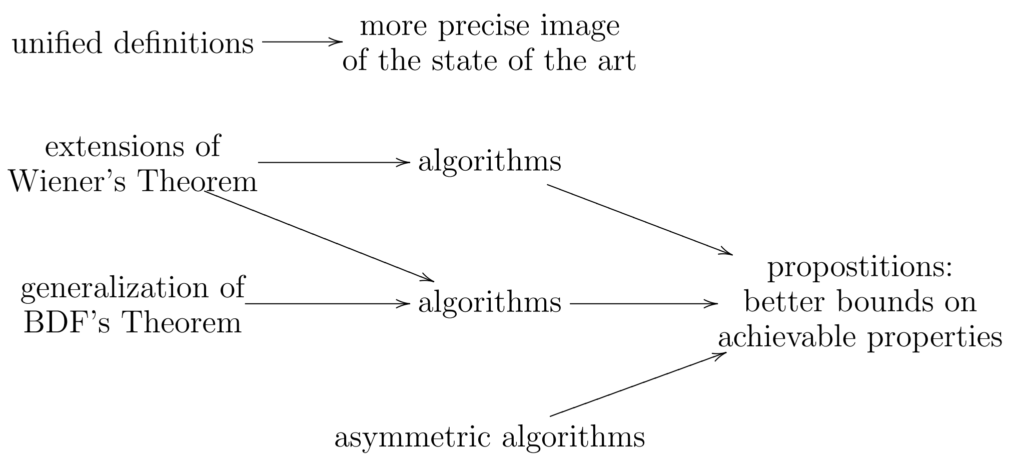

unified definitions
more precise image of the state of the art
extensions of Wiener's Theorem
algorithms
generalization of BDF's Theorem
algorithms
propostitions: better bounds on achievable properties
asymmetric algorithms

Figure 8–1: Relations between this work's results on backdoors.

As noted in the second part of the *Discussion* after Algorithm G4 (Figure 7–9, Subsection 7.1.4), an important open question is that the conditions of Theorem 2.2.11 are not sufficient to justify the intuition behind Assumption 2.2.26. It is on this assumption that the security of Algorithm G_4 is based. A different, perhaps related, theorem would be needed to provide the intuition that Theorem 2.2.11 appeared to.

Consequences. More concretely, these results have negative consequences on the trust which can be placed into cryptographic implementations which use public key generation. Three very different such examples are the following.

- PGP [PGP06]: as it is open source software, an embedded backdoor can be detected by the review of peers, but other users cannot if they view PGP as a black box.
- Microsoft: as it is closed source software, there is more reason to believe in issues such as the one of the NSAKEY, which is a parameter in Microsoft's Crypto API that is claimed to be a backup key, usable as a spare. However, mostly its name has given rise to conspiracy theories, as NSA can stand for *National Security Agency*, a United States governmental agency. [Sch99].
- Smart cards: as they are self-contained, small, black box devices, backdoored keys is a type of attack for which they are an ideal target (more complicated attacks are less likely).

Therefore, this call for *common criteria certification* involving revealing the key generation algorithm to a trusted third party [NSA06a], or RSA key generation with verifiable randomness [JG02]. The aim of the latter technique is to persuade the verifying party that the legitimate user has not weakened or reused its key. This is achievable via a simulated trusted third party and zero-knowledge proofs.

Theoretical importance. From a theoretical point of view, this thesis also contributes in terms of properties of backdoors. The proofs of these properties are constructive, referring to concrete backdoored key generators. The four new algorithms presented are hidden exponent methods where the running times are linear in the running time of the original algorithm. For all algorithms, except for the second one which plays an intuitively important role, the set of possible cheating keys can be made more diverse and better defined than in earlier algorithms. The first three algorithms are meant to develop intuition. The fourth backdoored key generator is our strongest result. It is also the best-defined one: cryptographic assumptions are made on explicit functions, actually RSA itself, instead of on an uninstanciated function, π_β, as it had been the case for the strongest previous results.

A general backdoored key generation algorithm building recipe is to use theorems that need a minimal amount of information on the key in order to break it, such as Theorems 7.1.12, 6.3.4 and 7.1.3. Then those redundant bits can be scrambled with random bits, which allows for a better diversity of the backdoored keys.

Applied importance. We improve on the proposals of [CS03], which already supersede the SETUP constructions, in a number of ways. Firstly, these proposals already provided a better timing analysis. Secondly, they allow more power to the distinguisher, power of which the description is repeated in this chapter in Subsection 7.1.1. The provided examples of SETUPs cannot allow these powers, which we deem reasonable. Thirdly, those examples use the random oracle model in their proofs, therefore calling for the use of strong cryptographic functions, which is both an extra assumption and a computational complexity cost.

Comparison with kleptography. In kleptographic vocabulary, our backdoors are closer to regular SETUPs. However, in regular SETUPs, asymmetric encryption is used by the attacker [YY97a, YY97b, Definition 1, Points 2 and 3], and this is indeed not our case. Yet, asymmetry implies uniformity, as defined in [YY05a, YY05b, Property 4], which in turn implies strong SETUPs (idem, Definition 1). Therefore, in the regular SETUPs definition, the attacker should be stated as using symmetric encryption, otherwise the regular definition collapses to the strong one.

Moreover, it appears that regular SETUPs have been dropped from study. In the latest kleptography papers (2005), regular SETUPs are no longer mentioned and the latest regular SETUP (for key generation) [YY96, Section 3] and [YY97a, Section 5] is a backdoored generator for which confidentiality does not hold because the attacker's key lengths are not secure. However, as mentioned above, symmetric backdoors beat other asymmetric ones in a number of ways, therefore symmetric backdoors are of significant interest. Therefore, kleptographic vocabulary has evolved in such a way that the best comparison for our backdoors is with the regular setups.

We do not analyze whether our backdoors are strong SETUPs. The strong SETUP approach remains valuable as it produces high key diversity and may generate accurate examples of indistinguishability proofs. Despite that, we believe that the ones of Young and Yung [YY05b] are unfinished. As discussed in Subsection 6.2.4, these proofs appear to be incomplete. In brief, a distinguisher D is assumed to non-negligibly distinguishes primes from the backdoored keys. The idea is to use D to invert the ECDDH key exchange. However there is no explanation as to how the shared secret is computed from D, when the probability that the oracle gives the solution is negligible.

Main open question. The main that this work highlights is whether it is possible to generate RSA symmetric backdoored keys with cardinality significantly greater than $2^{\frac{3}{2}k}$ (for either of the parameters n or e, that is over a total of 2^k possible honest keys for that parameter only). Via the principles we showed for constructing algorithms, this can be the same as asking whether it is possible to factor $n = pq$ when knowing significantly less than $k/2$ bits about its factorization.

Appendix A: Basic cryptology

A.1 Contemporary secure key lengths

NIST recommends the following secret (private) key lengths [NSA06b]. The breaking times are from 2001, in [RSAb], and are for the corresponding symmetric key size.

symmetric	RSA	elliptic curve	breaking time
56	512	112	less than 5 min.
80	1024	160	600 months
112	2048	224	
128	3072	256	10^{16} years
192	7680	384	
256	15360	512	

Table A.1: Contemporary secure key lengths with approximate equivalents.

In order to attain a comparable security to 80 bits of symmetric key, one uses 1024 bits of RSA private key or 160 bits of elliptic curve-based private key. Also, by year 2010, cryptosystems using the equivalent of 80 bits of symmetric key should be phased out.

For instance, the most recently solved challenges for elliptic curves were of 109 bits, in 2002 and 2004 [Sti06, p. 268]. The U.S. Government standards for top secret encryption requires 192 or 256 bits of AES (Advanced Encryption Standard) secret key [CNS06, p. 2].

For ElGamal over $\mathbb{Z}_p^*$, the private key length (the one of p) must be at least

300 digits (997 bits) and $p-1$ must be a multiple of a large prime [Sti06, p. 235]. ElGamal is slightly stronger than RSA, in the sense that it requires less bits of private key.

A.2 The ElGamal cryptosystem

The ElGammal (over $\mathbb{Z}_p^*$) cryptosystem and digital signature schemes [ElG85] are based on the generation of a random prime p such that $p-1$ is large enough (Section A.1) and has a large prime divisor [Sti06, p. 235], of a primitive element (generator) $\alpha \in \mathbb{Z}_p^*$, and of a random $\beta = \alpha^a \bmod p$, where a is generated randomly modulo $p-1$. The triplet (p, α, β) is made public.

```
Algorithm $G^{EG}$ [EG key generation]
1: Pick a random prime $p$ of appropriate size [1] .
2: repeat
3:     Pick a random $\alpha \in \mathbb{Z}_p^*$.
4: until $\forall$ prime $p_i$, s.t. $p_i | p-1$, it holds that $\left[\alpha^{(p-1)/p_i} \bmod p \neq 1\right]$.
5: Pick a random $0 \leq a \leq p-2$.
6: Compute $\beta = \alpha^a \bmod p$.
7: return $(p, \alpha, a, \beta)$.
```

Figure A.1: Honest key generation for the EG cryptosystem.

[1] Size is chosen consistently with respect to the values provided by Table A.1.

A.2.1 Number of keys

An element α of a cyclic group G is a generator if the order of α is $|G|$. In the cyclic group $\mathbb{Z}_p^*$, every element has some order, k, that divides $p-1 = |\mathbb{Z}_p^*|$, by an application of Lagrange's Theorem. Suppose that β has order k, so any other group element of the form $\gamma = \beta^i$, for some $0 \leq i \leq p-2$, has order

$$|\gamma| = \frac{k}{\gcd(k,i)}.$$

So there are exactly $\phi(k)$ elements of order k, one for each i that is coprime with k. Therefore, there are $\phi(p-1)$ possible generators. This argument may be referred to in Stinson's book [Sti06, Section 5.2.3].

Note that if primes p with known factorization of $p-1$ are generated, using a result from Kalai [Kal03], then Step 4 of G^{EG} can be sped up.

Therefore, the number of honest EG keys is:

$$\begin{aligned} \mathcal{N}_{eg} &= \#\{p\} \cdot \#\{\alpha\} \cdot \#\{a\} \\ &= p\,\phi(p-1)(p-1) \\ &\approx p^3 \end{aligned}$$

where $\phi(p-1) \approx p$, because $p-1$ is a multiple of a large prime. This is so in order to thwart existing polynomial-time algorithms that solve special cases of the discrete logarithm problem [Sti06, p. 235].

A.3 The ElGamal cryptosystem over elliptic curves

The following information is taken from Stinson [Sti06, Section 6.5], and especially from Section 6.5.4. Encryption with ElGamal cryptosystem over $\mathbb{Z}_p^*$ has a message to ciphertext expansion factor of two. Over elliptic curves (EC), this blowup is of about four, because each of the two points on an elliptic curve consists of two field elements. Nevertheless, we will immediately see how this factor of four is reduced to a factor of approximately two.

Definition A.3.1 ([Sti06], Definition 6.4) *Let $p > 3$ be a prime. The* **elliptic curve** *E defined by the equation $y^2 = x^3 + \xi x + \psi$ over $\mathbb{Z}_p$ is the set of solutions $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$, where $\xi, \psi \in \mathbb{Z}_p$ and $3\xi^3 + 27\psi^2 \not\equiv 0 \bmod p$. Along with E is defined $\mathcal{O} \in E$, the special point called the* **point at infinity***.*

Assumption A.3.2 (EC discrete logarithm problem) *The curve E is selected so that it has a cyclic subgroup H of prime order $ord(H) = h$, where the discrete logarithm problem is assumed to be computationally infeasible.*

For this, $h - 1$ is required to be large enough (Table A.1) and to have a large prime divisor.

For elliptic curves, the analog to multiplying two points in $\mathbb{F}_p^*$ is adding two points on E. The analog to raising a point in $\mathbb{F}_p^*$ to the k^{th} power is multiplying $P \in E$ by an integer k. By repeated doubling, $kP \in E$ can be computed in $\mathcal{O}(\lg k \ \lg^3 p)$ bit operations [Kob94, p. 178].

A more efficient variant of elliptic curve ElGamal is the Elliptic Curve Integrated Encryption Scheme (ECIES). The simplified version of ECIES presented in [Sti06, Figure 6.2] has a ciphertext blowup factor of about two. Simplified ECIES uses the standard technique of *point compression*, which reduces the storage required for points on an elliptic curve by a factor of about two, at the expense of increased time complexity. Point compression exploits the fact that there are two solutions for the coordinate y in the equation of E, and these solutions are additive inverses of one another. Therefore, storing a single bit is sufficient to compute y, when given x.

The following three algorithms are the components of ECIES, with the description of E understood to be public. They are given in Figure A.2 (key generation), Figure A.3 (encryption) and Figure A.4 (decryption).

The public key is an instance of the EC discrete logarithm problem of Assumption A.3.2. Suppose E, H where α generates H. Given $(\alpha, \beta) \in H \times H$, where $\beta = a\alpha$ with $a \in \mathbb{Z}_h^*$, the problem is to find a. This a is the corresponding private key. The points α and β lie on the curve E, so they have two coordinates each of size $|p|$.

```
Algorithm $G_{ECIES}$ key generation
1: Let $\alpha$ be a generator of $H$ s.t. $ord(\alpha) = h$ has a large prime divisor.
2: Pick a random $a \in \mathbb{Z}_h^*$.
3: Let $\beta = a\alpha$.
4: Return $(a, \alpha, \beta)$.
```

Figure A.2: Key generation for the ECIES cryptosystem.

The algorithm encrypts a message $m \in \mathbb{Z}_p^*$. The encrypted message (at Step 5 of the encryption) is denoted (γ, θ) and is such that $\gamma \in \mathbb{Z}_p \times \mathbb{Z}_2$ and $\theta \in \mathbb{Z}_p^*$. The pair (γ, θ) is the compressed version of $(k\alpha, k\beta m)$. Note that at Step 2, $x_o \neq 0$ such that x_o^{-1} exists for use in the decryption algorithm which follows (Step 2).

```
**Algorithm** $E^{ECIES}(m)$ [ECIES **encryption of** $m \in \mathbb{Z}_p^*$]
1: Pick random $k \in \mathbb{Z}_h^*$.
2: Set $(x_o, y_o) = k\beta$, with $x_o \neq 0$.
3: Set $\gamma = PointCompress(k\alpha)$.
4: Set $\theta = x_o m \bmod p$.
5: return $y = (\gamma, \theta)$.
```

Figure A.3: Encryption for the ECIES cryptosystem.

In the decryption algorithm, Step 1 is the compressed version of the computation of $k\beta = a(k\alpha)$. Step 2 is the compressed version of the computation of $m = (k\beta m)(k\beta)^{-1}$.

```
**Algorithm** $D^{ECIES}(y)$ [ECIES **decryption of** $(\gamma, \theta), \gamma \in \mathbb{Z}_p \times \mathbb{Z}_2, \theta \in \mathbb{Z}_p^*$]
1: Set $(x_o, y_o) = a \cdot PointDecompress(\gamma)$.
2: Set $m \equiv \theta \cdot x_o^{-1} \bmod p$.
3: return $m$.
```

Figure A.4: Decryption for the ECIES cryptosystem.

It is easy to verify that the scheme is correct: $\theta \cdot x_o^{-1} \equiv (x_o m)x_o^{-1} \bmod p \equiv m \bmod p$. The quantity x_o is known to the decryption party because, having the private key a, it can compute $a \cdot PointDecompress(\gamma) = ak\alpha = k\beta = (x_o, y_o)$.

A.3.1 Cardinality, probability distribution and ciphertext length

Theorem A.3.3 (Hasse's Theorem) *Let $\mathcal{N}_E$ be the number of points on an elliptic curve E defined over $\mathbb{F}_p$. Then*

$$|\mathcal{N}_E - (p+1)| \leq 2\sqrt{p}$$

where $|.|$ denotes the absolute value.

A statement and explanation of Hasse's Theorem can be found in [Kob94, p. 174].

Remark A.3.4 *On Step 1, k is of size within the range $|p + 1 \pm 2\sqrt{p}|$.*

This comes from that H is a subgroup of $E \subset \mathbb{Z}_p \times \mathbb{Z}_p$, and that the discrete logarithm is assumed to be intractable on H. Therefore, one would aim to set E such that H is as close to being of the size of E as possible.

Example A.3.5 Suppose that the minimum secure length of p as of Table A.1 is used, which is $|p| = 160$, and p's most significant bit is 1. Therefore, k is of size of about $|p + 1 \pm 2\sqrt{p}| \in [159, 161]$. $\square$

The following proposition is the EG version of Proposition 2.2.27 for RSA.

Proposition A.3.6 *The* ECIES *encryption of m is pseudo-random, assuming the average-case hardness of the discrete logarithm on elliptic curves.*

proof sketch: This property is related to *semantic security* or *indistinguishability of encryptions* [GM84], usually denoted IND. Tsiounis and Yung have shown semantic security for EG [TY98].

However, the indistinguishability property sufficient for this work's results is that the output of some encryption function be pseudo-random. This appears to be a property stronger than indistinguishability of encryptions. Nevertheless, Phan and Pointcheval have shown an equivalence between indistinguishability of encryptions and pseudo-randomness [PP04]. ■

Remark A.3.7 *The* ECIES *encryption of a message m has length*

$$2 \cdot |m| + 1 = 2 \cdot |p| + 1.$$

Appendix B: Computational number theory

A number of theorems from computational number theory are useful to compress or scramble information. This applies especially well to backdoors in the key generation of cryptosystems that are based on number-theoretical properties, such as RSA. The entirety of the theorems listed in this appendix are also in the main text. They are repeated here for the purpose of forming a compilation of useful theorems.

B.1 Lattice basis reduction theorems

An approximate p allows the factoring of n if the additive error is on its lower-half bits.

Theorem B.1.1 ([Cop96], Corollary 2) *Given an approximation ρ of p such that $|p-\rho| \leq n^{1/4}$, it is possible to factor n in time polynomial in k, that is,* $\lg(n)$.

Refer to Corollary 6.1.8 in the main text.

Theorem B.1.2 ([Cop96], Theorem 3) *Let $f(x, y)$ be a polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately, and assume the coefficients of f are relatively prime as a set. Let X, Y be bounds on the desired solutions*

x_o, y_o. Define $\tilde{f} := f(Xx, Yy)$ and let D be the absolute value of the largest coefficient of $\tilde{f}$. If $XY < D^{2/3\delta}$, then in time polynomial in $\lg D$ and 2^δ, we can find all integer pairs (x_o, y_o) with $f(x_o, y_o) = 0$, $|x_o| < X$,$|y_o| < Y$.

Refer to Theorem 6.1.7 in the main text.

The proof of Theorem B.1.2 rests on the LLL lattice basis reduction. It yields Theorem B.1.3, by taking $p_o \equiv p \bmod \beta$, $q_o \equiv q \bmod \beta$ and letting $f(x,y) = (\beta x + p_o)(\beta y + q_o) - n$.

Theorem B.1.3 ([BDF98], Corollary 2.2) *Given $\beta \geq 2^{k/2}$ and $p_0 \equiv p \bmod \beta$, it is possible to factor $n = pq$ in time polynomial in k, denoted by $T(k)$.*

Refer to Theorem 7.1.12 in the main text.

Theorem B.1.4, as in Howgrave-Graham [HG01, p. 52], is similarly derived from Theorem B.1.2 by taking $p_o' = kp + x_o$, $q_o' = kq + y_o$, so that $(p_o' - x_o)(q_o' - y_o) = k^2 n$, and letting $f(x,k) = (p_o' - x)(q_o' x - n) - xk^2 n$.

Theorem B.1.4 (Corollary of Theorem 6.1.7) *Given $p_o' = kp + x_o$ for $|k| < n^{1/2}$ and $|x_o| < n^{1/4}$, it is possible to factor $n = pq$ in time polynomial in k.*

Refer to Corollary 6.1.9 in the main text.

B.2 Wiener's Theorem

Wiener's Theorem [Wie90] is useful in many versions. Suppose that $n = pq$ is an RSA modulus and denote a public key as (n, ϵ) and the corresponding private key as δ. The parameter κ is such that $\epsilon\delta - \kappa\phi(n) = 1$, where ϕ is the Euler totient function.

Theorem B.2.1 (Wiener's low decryption exponent attack) *Any (n, ϵ) with $3\delta < n^{1/4}$ efficiently yields the values of κ and δ.*

Refer to Theorem 6.3.1 in the main text.

In the following theorem, recall that $\epsilon\rceil^t$ denotes the t most significant bits of ϵ and $\epsilon\rfloor_t$, its t least significant ones.

Theorem B.2.2 ([BDF98], Theorem 4.6) *Let $t \in [1, ..., k]$ and $\delta \in [2^t, ..., 2^{t+1}]$. Given (n, δ), $\epsilon\rceil^t$ and $\epsilon\rfloor_{k/2}$, one can factor n efficiently.*

Refer to Theorem 6.3.4 in the main text.

Theorem B.2.3 (Extension of Wiener's Theorem) *Let $\alpha \leq 1/4$, $\ell_1 < \frac{1}{3}$ and ℓ_2 s.t. $\ell_1^2(\ell_2 + 3) \leq \frac{1}{3}$. Any (n, e) related to (n, ϵ), with concealed ϵ, that satisfies $|e - \epsilon| < \ell_2\, n^{1-2\alpha}$, and $\delta < \ell_1\, n^{\alpha}$ efficiently yields the values of κ and δ.*

Refer to Theorem 7.1.3 in the main text.

B.3 Other theorems

Theorem B.3.1 ([Red96], Theorem 3.16) *A necessary and sufficient condition for* $y^2 \equiv b \pmod{\beta}$*, to be solvable for* y*, where* $\gcd(b, \beta) = 1$*, is that* b *be a quadratic residue of all odd prime divisors of* β *and that if* $2||\beta$*, then* b *is odd, if* $4||\beta$*, then* $b \equiv 1 \pmod{4}$*, and if* $8|\beta$*, then* $b \equiv 1 \pmod{8}$*.*

Refer to Theorem 7.1.11 in the main text.

References

[ACK] G. Arboit, C. Crépeau, and R.A. Kazmi. Backdoors for ElGamal and public key backdoors for RSA and ElGamal. To be submitted.

[ACS] G. Arboit, C. Crépeau, and A. Slakmon. An extension of Wiener's Theorem and applications to RSA backdoors. To be submitted.

[And93] R. Anderson. A practical RSA trapdoor. *Electronics Letters*, 29(11):995, 1993.

[AR01] G. Arboit and J.-M. Robert. From fixed-length messages to arbitrary-length messages practical RSA signature padding schemes. *Lecture Notes in Computer Science*, 2020:44–51, 2001.

[Arb] G. Arboit. Measures for backdoors in public key generation. To be submitted.

[BD00] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $n^{0.292}$. *IEEE Transactions on Information Theory*, 46:1339–1349, 2000.

[BDF98] D. Boneh, G. Durfee, and Y. Frankel. An attack on RSA given a small fraction of the private key bits. *LNCS 1514, Proceedings of Advances in Cryptology - AsiaCrypt '98*, pages 25–34, 1998.

[BG85] M. Blum and S. Goldwasser. An efficient probabilistic public key encryption scheme which hides all partial information. *Proceedings of Advances in Cryptology - CRYPTO '84*, pages 289–299, 1985.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. *Lecture Notes in Computer Science*, 1462:1–12, 1998.

[BM04] J. Blömer and A. May. A generalized wiener attack on RSA. *Lecture Notes in Computer Science 2947, PKC 2004*, 2004.

[BP97] N. Barić and B. Pfitzmann. Collision-free accumulators and Fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, Lecture Notes in Computer Science Vol. 1233*, pages 480–494. Springer, 1997.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. *Lecture Notes in Computer Science*, 1070:399–416, 1996.

[BS96] E. Bach and J. Shallit. *Algorithmic number theory, volume 1: efficient algorithms*. MIT Press, Cambridge, Massachusetts, 1996. URL: `http://www.math.uwaterloo.ca/ shallit/ant.html`.

[BV98] D. Boneh and R. Venkatesan. Breaking rsa may not be equivalent to factoring. In *EUROCRYPT*, pages 59–71, 1998.

[CCKL01] J. C. Cha, J. H. Cheon, K. H. Ko, and S. J. Lee. Tutorial on braid cryptosystem. *PKC*, 2001.

[CFPR96] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter. Low-exponent RSA with related messages. In *Advances in Cryptology - Eurocrypt '96, LNCS 1070*, pages 1–9. Springer, 1996.

[Cho04] J. Cho. Ten years of RSA cheating cryptosystems, 2004.

[CKN00] J.-S. Coron, F. Koeune, and D. Naccache. From fixed-length to arbitrary-length RSA padding schemes. In *Advances in Cryptology - ASIACRYPT '00*, pages 90–96. Springer, 2000.

[CNS06] Committee on National Security Systems (CNSS). CNSS Policy no. 15, Fact sheet no. 1. http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf, 2006.

[Coc73] C. C. Cocks. A note on non-secret encryption, 1973. http://www.cesg.gov.uk/site/publications/media/notense.pdf.

[Cop96] D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *LNCS 1070, Advances in Cryptology - EuroCrypt '96*, pages 178–189. Springer, 1996.

[Cop97] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10:233–260, 1997.

[CS03] C. Crépeau and A. Slakmon. Simple backdoors for RSA key generation. *Lecture Notes in Computer Science 2612, CT-RSA*, pages 402–415, 2003.

[Dav82] G. I. Davida. Chosen signature cryptanalysis of the RSA (mit) public key cryptosystem. Technical Report TR-CS-82-2, Departement of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, 1982.

[Des88] Y. Desmedt. Abuses in cryptography and how to fight them. In *LNCS 403, CRYPTO*, pages 375–389, 1988.

[DF91] D.S. Dummit and R.M. Foote. *Abstract Algebra.* Prentice Hall, 1991.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.

[dis02] distributed.net completes rc5-64 project (list announcement). http://www1.distributed.net/pressroom/news-20020926.txt, 25 September, 2002.

[Dod06] B. Dodson. Ps on ecm paper. http://www.loria.fr/ zimmerma/records/p67, 2006.

[dW02] B. de Weger. Cryptanalysis of RSA with small prime difference. *Applicable Algebra in Engineering, Communication and Computing*, (13):17–28, 2002.

[ElG85] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[Ent] Entrust: What is a PKI? http://www.entrust.com/pki.htm.

[FOPS01] E. Fugisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSAOAEP Is Secure under the RSA Assumption. In *Advances in Cryptology - Crypto'01*, pages 260–274, 2001.

[GHR99] R. Gennaro, S. Halevi, and T. Rabin. Secure Hash-and-Sign Signatures without the Random Oracle. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, Vol. 1592 of Lecture Notes in Computer Science*, pages 123–139. Springer, 1999. http://www.research.ibm.com/security/ghr.ps.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Jour. of Computer and System Science*, 28(2):270–299, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. March 23, 1995 revision.

[GMT82] S. Goldwasser, S. Micali, and P. Tong. Why and how to establish a private code on a public network. In *Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 134–144, 1982.

[Hås88] J. Håstad. Solving simultaneous modular equations of low degree. *SIAM J. Comput.*, 17(2):336–341, 1988.

[HG01] N. Howgrave-Graham. Approximate integer common divisors. *Lecture Notes in Computer Science 2146, Cryptography and Lattices*, pages 51–66, 2001.

[HN98] J. Håstad and M. Näslund. The security of individual RSA bits. In *FOCS*, pages 510–521, 1998.

[HW79] G.H. Hardy and E.M. Wright. *An introduction to the theory of numbers.* Oxford University Press, 1979.

[JG02] A. Juels and J. Guajardo. RSA key generation with verifiable randomness. In D. Naccache and P. Paillier, editors, *Public Key Cryptography 2002, Vol. 2274 of Lecture Notes in Computer Science*, pages 357–374. Springer, 2002. http://www.rsasecurity.com/rsalabs/node.asp?id=2041.

[JPV00] M. Joye, P. Paillier, and S. Vaudenay. Efficient generation of prime numbers. In *CHES '00: Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, pages 340–354. Springer-Verlag, 2000.

[Kal93] B.S. Kaliski Jun. Anderson's RSA trapdoor can be broken. *Electronics Letters*, 29(15):1387–1388, 1993.

[Kal03] A. Kalai. Generating random factored numbers, easily. *Journal of Cryptology*, 16(4):287–289, 2003. Also at SODA 2002.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[Kob94] N. Koblitz. *A Course in Number Theory and Cryptography.* Graduate Texts in Mathematics 114. New York: Springer-Verlag, 1994.

[KP99] S. Katzenbeisser and F. Petitcolas, editors. *Information hiding techniques for steganography and digital watermarking.* Artech House Books, 1999.

[LLL82] A.K. Lenstra, H.W. Lenstra Jr., and L. Lovasc. Factoring polynomials with rational coefficients. *Math. Annalen*, (261):513–534, 1982.

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, 1996.

[May04] A. May. Computing the RSA secret key is deterministic polynomial time equivalent to factoring. In *CRYPTO*, pages 213–219, 2004.

[Mer78] R.C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, April 1978.

[MH78] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE-IT*, IT-24:525–530, 1978.

[Mil75] G. L. Miller. Riemann's hypothesis and tests for primality. In *STOC '75: Proceedings of seventh annual ACM symposium on Theory of computing*, pages 234–239, New York, NY, USA, 1975. ACM Press.

[Mil85] V. Miller. Use of elliptic curves in cryptography. *Lecture Notes in Computer Science, CRYPTO 85*, 1985.

[Mis98] J.-F. Misarsky. How (Not) to Design Signature Schemes. In *Proceedings of PKC '98, Lecture Notes in Computer Science Vol. 1431.* Springer, 1998.

[MvV01] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* Discrete Mathematics and its Applications Vol. 6. CRC Press, 2001.

[NSA06a] National Security Agency (NSA). national information assurance partnership (niap). http://www.nsa.gov/ia/industry/niap.cfm, 2006.

[NSA06b] National Security Agency (NSA). the case for elliptic curve cryptography. http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm, August 2006.

[PGP06] The international PGP home page (PGPI). http://www.pgpi.org/, 2006.

[PKC03] Public-key cryptography standards (pkcs) #1: RSA cryptography specifications version 2.1. http://www.ietf.org/rfc/rfc3447.txt, 2003.

[PP04] D.H. Phan and D. Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *Selected Areas in Cryptography (SAC), LNCS 3357*, pages 182–197. Springer, 2004.

[Rab79] M.O. Rabin. Digital signatures and public-key functions as intractable as factorization. *Technical Report MIT/LCS/TR-212, MIT Lab. for Computer Science, Cambridge, MA*, January 1979.

[Red96] D. Redmond. *Number Theory: An Introduction, Monographs and Textbooks in Pure and Applied Mathematics no. 201.* Marcel Dekker, 1996.

[Rot01] R. Roth. A History of Lagrange's Theorem on Groups. *Mathematics Magazine*, pages 99–108, 2001.

[RS86] R. L. Rivest and A. Shamir. Efficient factoring based on partial information. In *Advances in Cryptology - EuroCrypt '85*, pages 31–34, 1986.

[RS03] K. Rubin and A. Silverberg. Torus-based cryptography. *Proceedings of Advances in Cryptology - CRYPTO*, pages 349–365, 2003.

[RS06] A. Rostovtsev and A. Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. http://eprint.iacr.org/.

[RSAa] RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1. The Crypto FAQ: Index page: http://www.rsasecurity.com/rsalabs/node.asp?id=2152.

[RSAb] RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1. The Crypto FAQ: Section 2.3.2: What is a one-way function?: http://www.rsasecurity.com/rsalabs/node.asp?id=2088.

[RSAc] RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1. The Crypto FAQ: Section 6.2.4: What is Clipper?: http://www.rsasecurity.com/rsalabs/node.asp?id=2318.

[RSA78] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining. *Communications of the ACM*, 21:120–126, 1978.

[RSA06] The RSA laboratories secret-key challenge: Status and prizes. http://www.rsasecurity.com/rsalabs/node.asp?id=2103, August 2006.

[Sch99] B. Schneier. Crypto-gram newsletter, September 15, 1999.

[Sha82] A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *FOCS'82*, 1982.

[Shp04] I. E. Shparlinski. On the uniformity of distribution of the decryption exponent in fixed encryption exponent RSA. *Information processing letters*, 92(3):143–147, 2004.

[Sim83] G.J. Simmons. The prisoners' problem and the subliminal channel. In *CRYPTO*, pages 51–67, 1983.

[Sim85] G.J. Simmons. A secure subliminal channel (?). In *CRYPTO*, pages 33–41, 1985.

[Sla00] A. Slakmon. Sur des méthodes et algorithmes de factorisation et leur application en cryptologie. Master's thesis, Université de Montréal, dépt. IRO, 2000.

[SPW06] Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. On the provable security of an efficient rsa-based pseudorandom generator. Cryptology ePrint Archive, Report 2006/206. To appear in Asiacrypt'06, 2006.

[Sti06] D. R. Stinson. *Cryptography Theory and Practice, 3rd edition.* Chapman & Hall/CRC Press, 2006.

[TY98] Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. *Public Key Cryptography '98, Lecture Notes in Computer Science*, 1431:117–134, 1998.

[Vau] S. Vaudenay. Private e-mail communication. 2 May 2001.

[Way97] P. Wayner. British document outlines early encryption discovery. *The New York Times*, December 24, 1997.

[Wei07] E. Weisstein. "fermat's factorization method.". From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/FermatsFactorizationMethod.html, 2007.

[Wie90] M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36:553–558, 1990.

[WIF06] World integer factorization center (wifc). http://www.asahi-net.or.jp/ KC2H-MSM/mathland/matha1/index.htm, August 2006.

[YY96] A. Young and M. Yung. The dark side of “black-box” cryptography, or: Should we trust Capstone? *Lecture Notes in Computer Science 1109, Crypto ’96*, pages 89–103, 1996.

[YY97a] A. Young and M. Yung. Kleptography: Using cryptography against cryptography. *Lecture Notes in Computer Science 1233, EuroCrypt ’97*, pages 62–74, 1997.

[YY97b] A. Young and M. Yung. The prevalence of kleptographic attacks on discrete-log based cryptosystems. *Lecture Notes in Computer Science 1294, Crypto ’97*, pages 264–276, 1997.

[YY05a] A. Young and M. Yung. Malicious cryptography: Kleptographic aspects. *Lecture Notes in Computer Science 3376, CT-RSA 2005*, pages 7–18, 2005.

[YY05b] A. Young and M. Yung. A space efficient backdoor in RSA and its applications. *Selected Areas in Cryptography – SAC 2005*, 2005.

Index

Index