Faculty of Science
Final Examination

Computer Science COMP-547A
Cryptography and Data Security

Examiner: Prof. Claude Crépeau **Date:** Dec 21st, 2006

Associate Examiner: Prof. David Avis **Time:** 14:00 – 17:00
Room: ARTS 150

INSTRUCTION:
- This examination is worth 50% of your final grade.
- The total of all questions is 110 points.
- Each question heading contains (in parenthesis) a list of values for each sub-questions.
- This is an **open book** examination. All documentation is permitted.
- Faculty standard calculator permitted only.
- The exam consists of 6 questions on 5 pages, title page included.

Suggestion: read all the questions and their values before you start.

Question 1. Schnorr Identification (5+5+5* points)

Let p,q,α,t be the public parameters of Schnorr identification scheme as published by the **TA**. Let $v=\alpha^a \bmod p$ be a Alice s public key. Consider an algorithm **A** such that

> **PROC A**
> **INPUT:** p,q,α,t,v
> **OUTPUT:** γ,y_1,y_2
>
> with $y_i = \log_\alpha \gamma v^{-r_i}$ for two arbitrary r_1,r_2 in the range $0 < r_1,r_2 < 2^t$.

• Show how to use algorithm **A** to cheat Schnorr s scheme with probability at least $2/2^t$.

• Show how to use algorithm **A** to compute v s discrete log with probability at least $1/4^t$.

(*) Show how to use algorithm **A** to compute v s discrete log with probability at least $1/2^t$.

Reminder

Protocol 9.8: SCHNORR IDENTIFICATION SCHEME

1. Alice chooses a random number, k, where $0 \leq k \leq q-1$, and she computes $\gamma = \alpha^k \bmod p$. She sends $\mathbf{Cert}(Alice)$ and γ to Bob.
2. Bob verifies Alice's public key, v, on the certificate $\mathbf{Cert}(Alice)$. Bob chooses a random challenge r, $1 \leq r \leq 2^t$, and he sends r to Alice.
3. Alice computes $y = k + ar \bmod q$ and she sends the response y to Bob.
4. Bob verifies that $\gamma \equiv \alpha^y v^r \pmod p$. If so, then Bob "accepts"; otherwise, Bob "rejects."

Question 2. Server-aided RSA signatures (4+4+7 points)

Let $n=pq$ be the product of two large primes $p \equiv q \equiv 2 \pmod 3$. Let $(n,e=3)$ be the public key of Bob s **RSA** digital signature scheme. Let $d=3^{-1} \bmod \phi(n)$ be Bob s private key. Suppose Bob is a low efficiency processor who trusts a very efficient server Ben enough to give him his private key d.

• Explain why choosing exponent 3 is a better choice than an arbitrary **RSA** exponent.

• Explain why we requested $p \equiv q \equiv 2 \pmod 3$.

• Show how to use **RSA** s multiplicative property in such a way that Bob can get Ben to sign messages for him, but in a way that discloses no information about the actual message to Ben. In other words, if M is the message Bob wants to sign and M is the message signed by Ben then $I(M;M\ )=0$. Explain how little computation Bob needs to do.

Question 3. Short and *Sweet* (5+5+5+6+5 points)

(justify briefly your answers)

(a)
Explain how a deterministic digital signature scheme resistant to existential forgeries is analogous to a pseudo-random function generator but cannot possibly be one !

(b)
Define a family of functions $F_k\colon\{0,1\}^{56}\to\{0,1\}^{64}$ as $F_k(x) = \mathbf{DES}_x(k)$. Explain how you can very efficiently discover that the family $\{F_k\}_{k\in\{0,1\}^{64}}$ is not pseudo-random.

(c)
Define a hash function $H\colon\{0,1\}^{256}\to\{0,1\}^{128}$ as $H(x_1\|x_2) = \mathbf{AES}_{x_1}(x_2)$. Show that this hash function is useless for cryptographic purposes because the **Preimage** problem is easy.

(d)
Identify two finite fields where the number of elements is a 1025-bit number.
(If you cannot find explicit examples then, for partial credit, tell us how to compute them.)

(e)
Remember the algorithms to verify primitive roots and irreducible polynomials.
Why is factoring q-1 an important issue in the first but factoring n in the second is not ?

Reminder

Algorithm 2.1 (*Primitive*(q))

1: *Let* $l_1, l_2, \ldots, l_k$ *be the prime factors of* $q-1$ *and* $m_i = \frac{q-1}{l_i}$ *for* $1\le i\le k$,

2: **REPEAT**

3: *pick a random non-zero element* g *of* $\mathcal{F}_q$,

4: **UNTIL** $g^{m_i}\neq 1$ *for* $1\le i\le k$,

5: **RETURN** g.

Algorithm 2.3 (**Rabin** *Irr*(p,n))

1: *let* $l_1, l_2, \ldots, l_k$ *be the prime factors of* n *and* $m_i = n/l_i$ *for* $1\le i\le k$,

2: **REPEAT**

3: *pick a random polynomial* $h(x)$ *of degree* $n-1$ *over* $\mathcal{F}_p$,
$g(x)\leftarrow x^n + h(x)$,

4: **UNTIL** $x^{p^n} \bmod g(x) = x$ *and* $gcd(g(x), x^{p^{m_i}} - x) = 1$ *for* $1\le i\le k$,

5: **RETURN** g.

Question 4. Double RSA signature (12 points)

Let $n=pq$ be the product of two large primes. Let $(e,d),(e\ ,d\ )$ be two pairs of **RSA** public/private exponents mod n. Consider the **DRSA (double-RSA)** signature scheme of a message m to be $(m,\ m^d \bmod n,\ m^d \bmod n)$. Analyze the impact on existential-forgery attacks on **RSA** signatures, in the context of this improved way of signing messages.

Question 5. AES key schedule (12 points)

Consider the **AES** key schedule for key of 128 bits. Currently the key schedule produces 44 words (32-bits each) such that the first 4 words are a copy of the original 128-bit key. The next 40 words are produced by the forward key-schedule algorithm.

Show how to modified the key schedule algorithm in such a way that the original key is the last **4** words and the rest of the schedule produces the unique sequence that ends with these **4** words. In other words, give an explicit algorithm to compute the **AES** key schedule backwards. (Assume you have inverse functions RotWordInv and SubWordInv.)

Reminder

Algorithm 3.6: KEYEXPANSION(key)

external ROTWORD, SUBWORD
$RCon[1] \leftarrow 01000000$
$RCon[2] \leftarrow 02000000$
$RCon[3] \leftarrow 04000000$
$RCon[4] \leftarrow 08000000$
$RCon[5] \leftarrow 10000000$
$RCon[6] \leftarrow 20000000$
$RCon[7] \leftarrow 40000000$
$RCon[8] \leftarrow 80000000$
$RCon[9] \leftarrow 1B000000$
$RCon[10] \leftarrow 36000000$
for $i \leftarrow 0$ **to** 3
  do $w[i] \leftarrow (key[4i], key[4i+1], key[4i+2], key[4i+3])$
for $i \leftarrow 4$ **to** 43
  do $\begin{cases} temp \leftarrow w[i-1] \\ \textbf{if } i \equiv 0 \pmod 4 \\ \quad \textbf{then } temp \leftarrow \text{SUBWORD}(\text{ROTWORD}(temp)) \oplus RCon[i/4] \\ w[i] \leftarrow w[i-4] \oplus temp \end{cases}$
return $(w[0], \ldots, w[43])$

Question 6. Merkle-Damgård iterated hash (5+5+5+10+5* points)

Remember the **Merkle-Damgård** hash function, based on a fixed size ($\{0,1\}^{m+t} \rightarrow \{0,1\}^m$) compression function named **compress**.

• Show that for all n, d is non-negative and $d<t$. Show that the binary representation of d will always fit in t-1 bits (which is the size of y_{k+1}).

• Find an alternative coding scheme for d such that at most one bit of y_{k+1} is a one.

• Argue that the d least significant bits of y_k and the m most significant bits of z_1 could be any fixed patterns.

• Let $t<2^m$. Let w := "the binary representation of d on m bits". Prove that if we set

$$z_1 := w \,||\, 0 \,||\, y_1$$

(and the last block is y_k, not y_{k+1}) then the security properties remain unaffected.

(*) Explain why an arbitrary compression function mapping $\{0,1\}^{m+2^m} \rightarrow \{0,1\}^m$ cannot be seriously considered collision-resistant.

Reminder

Algorithm 4.6: MERKLE-DAMGÅRD(x)

external compress
comment: compress : $\{0,1\}^{m+t} \rightarrow \{0,1\}^m$, where $t \geq 2$

$n \leftarrow |x|$
$k \leftarrow \lceil n/(t-1) \rceil$
$d \leftarrow k(t-1) - n$
for $i \leftarrow 1$ **to** $k-1$
  do $y_i \leftarrow x_i$
$y_k \leftarrow x_k \,\|\, 0^d$
$y_{k+1} \leftarrow$ the binary representation of d
$z_1 \leftarrow 0^{m+1} \,\|\, y_1$
$g_1 \leftarrow$ **compress**(z_1)
for $i \leftarrow 1$ **to** k
  do $\begin{cases} z_{i+1} \leftarrow g_i \,\|\, 1 \,\|\, y_{i+1} \\ g_{i+1} \leftarrow \textbf{compress}(z_{i+1}) \end{cases}$
$h(x) \leftarrow g_{k+1}$
return $(h(x))$