Computer Science 308-547A
Cryptography and Data Security

Claude Crépeau

These notes are, largely, transcriptions by Anton Stiglic of class notes from the former course *Cryptography and Data Security (308-647A)* that was given by prof. Claude Crépeau at McGill University during the autumn of 1998-1999. These notes are updated and revised by Claude Crépeau.

15 El Gamal

This scheme was invented by El Gamal [**?**]. Its security is based on the DH and DL problems.

Algorithm 15.1 (El Gamal key generation)

1: *Pick a large random prime* p*.*

2: *Pick a generator* α *of* $\mathbb{Z}_p^*$ *and a random integer* a*,* $1 \le a \le p-2$*.*

3: $\beta \leftarrow \alpha^a \bmod p$*.*

4: RETURN $K_e = (p, \alpha, \beta)$ *and* $K_d = a$*.*

Algorithm 15.2 (El Gamal public-key encryption)

1: *Pick a random integer* k*,* $1 \le k \le p-2$*.*

2: $\gamma \leftarrow \alpha^k \bmod p$*,* $\delta \leftarrow m \cdot \beta^k \bmod p$*.*

3: RETURN$c = (\gamma, \delta)$*.*

Decryption: $m \leftarrow \delta(\gamma)^{-a} \bmod p$.
Proof that $d(e(x)) = x$:

$$\begin{aligned} d(\gamma, \delta) &\equiv \delta \cdot (\gamma)^{-a} \bmod p \\ &\equiv m \cdot \beta^k (\alpha^k)^{-a} \bmod p \\ &\equiv m \cdot \beta^k (\beta)^{-k} \bmod p \\ &\equiv m \bmod p. \end{aligned}$$

Theorem 15.1 *Random integers* k *must be used if no information about the cleartexts is to be revealed.*

Proof. Say we have

$$\begin{aligned} e(m_1, k) &= (\alpha^k \bmod p, \underbrace{m_1 \cdot \beta^k \bmod p}_{\delta_1}) \text{ and} \\ e(m_2, k) &= (\alpha^k \bmod p, \underbrace{m_2 \cdot \beta^k \bmod p}_{\delta_2}) \end{aligned}$$

we can then compute $\delta_1/\delta_2 = m_1/m_2 \bmod p$.

Theorem 15.2 *The security of the El Gamal System is based on the DH problem.*

Proof. Denote $O_{ElGamal}$ to be an oracle for decrypting El Gamal encrypted messages, given $p, \alpha, \alpha^a, \alpha^k, m \cdot \alpha^{ak}$ and O_{DH} an oracle for solving the DH (that is $DH(p, \alpha, \alpha^a, \alpha^b)$ gives α^{ab}).

- ($O_{ElGamal}$ from O_{DH}) Compute α^{ak} using O_{DH}. We then have $m \leftarrow (\alpha^{ak})^{-1} \cdot \delta$.

- (O_{DH} from $O_{ElGamal}$) Pick a random $\delta \in \mathbb{Z}_p$. Compute $O_{ElGamal}(p, \alpha, \alpha^a, \alpha^b, \delta)$, which gives m such that $m \cdot \alpha^{ab} = \delta$. We have $\alpha^{ab} \leftarrow \delta \cdot m^{-1} \bmod p$.

15.1 Generalizing El Gamal

We described the El Gamal system in a group $\mathbb{Z}_p^*$, but it can be generalized to work in any finite cyclic group G. It's security is then *based* on the DL problem of that particular group. Examples of groups in which the DL problem is believed to be hard and in which operations can be efficiently executed are

$$\mathbb{Z}_p^*, \mathcal{F}_{2^m}^*, \mathcal{F}_{q^m}^*, \mathbb{Z}_{pq}^*$$

.

Theorem 15.3 *In the case of $\mathbb{Z}_N^*$ where $N = pq$ and $p \equiv q \equiv 3 \pmod 4$, the DL problem is as hard as FACTORING.*

Note: for the case of $\mathbb{Z}_N^*$ where $N = pq$, it is recommended to pick p and q in such a way that $p - 1$ and $q - 1$ do not have small factors, so as to guard against *Pollard's* factoring algorithm.

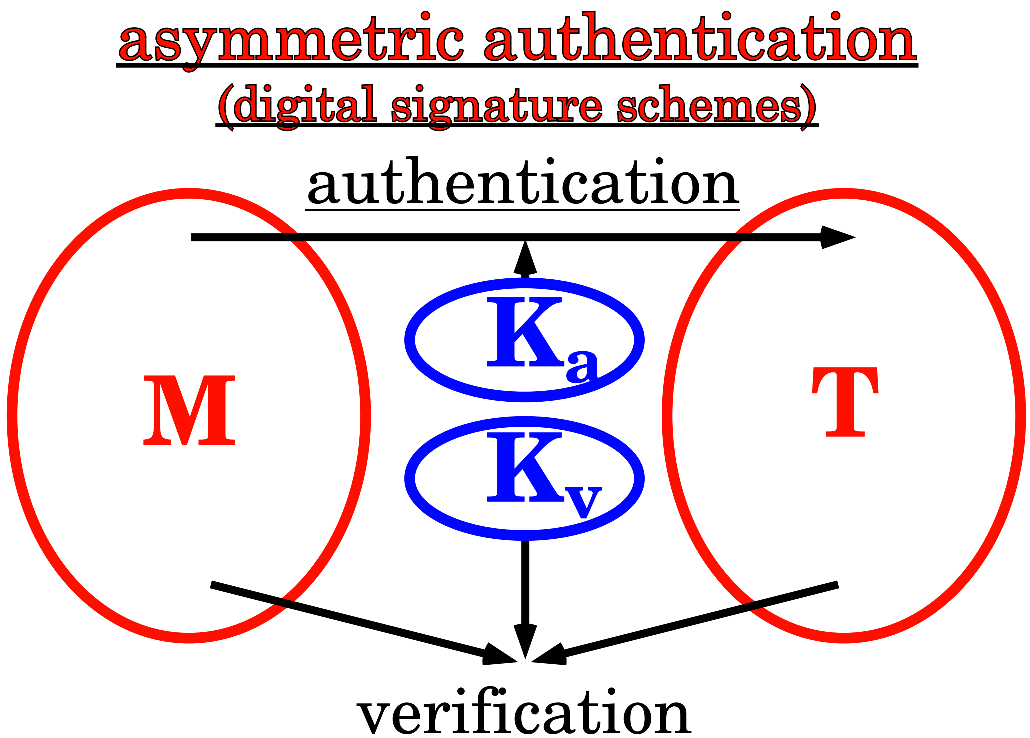
16 Digital signatures

A digital signature scheme allows *Alice* to compute a *signature* s for a message m in a way that Bob, and others, can verify that s was in fact computed by *Alice* and no one else.
Formally, a digital signature scheme is defined as follows:

Definition 16.1 *Let $\mathcal{M}$ be a finite set of messages and $\mathcal{T}$ a finite set of digital signatures such that for each $(k_a, k_v) \in \mathcal{K}$, there is a signing algorithm sig_{k_a} and a corresponding verification algorithm ver_{k_v} such that $sig_{k_a} : \mathcal{M} \to \mathcal{T}$ and $ver_{k_v} : \mathcal{M} \times \mathcal{T} \to \{true, false\}$ are polynomial-time computable functions and*

$$ver_{k_v}(m, y) = \begin{cases} true & : \quad if\ y = sig_{k_a}(x) \\ false & : \quad if\ y \neq sig_{k_a}(x) \end{cases}$$

A major difference between an authentication scheme and a signature scheme is that in an authentication scheme where *Alice* authenticates herself to *Bob*, *Bob* can "fake" *Alice*'s authentication for any message.



16.1 RSA signature scheme

The RSA cryptographic scheme can be directly used as a signature scheme: the decryption function is used as the signature function and the verification function is obtained by comparing the message with the encryption of the signature.

16.2 ElGamal signature scheme

We use the same keys as in the ElGamal encryption scheme, that is we have $\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p),\ \alpha \text{ a generator of } \mathbb{Z}_p^*\}$, p, α and β are public, a is kept secret.
Unlike RSA, the functions for the ElGamal signature scheme are not identical to those of the ElGamal encryption scheme. The functions are constructed to try to make forgery difficult.

Algorithm 16.1 (ElGamal signature)

1: *Pick a random k such that $1 \leq k \leq p-2$ and gcd$(k, p-1) = 1$.*

2: $\gamma \leftarrow \alpha^k \bmod p$, $\delta \leftarrow (x - a\gamma) \cdot k^{-1} \bmod p - 1$.

3: **RETURN** $s = (\gamma, \delta)$.

Verification:

$$Ver_K(x, \gamma, \delta) = true \iff \beta^\gamma \gamma^\delta = \alpha^x \bmod p$$

If the signature was constructed correctly, then the verification will succeed since

$$\begin{aligned} \beta^\gamma \gamma^\delta &= \beta^\gamma (\alpha^k)^\delta \bmod p \\ &= \beta^\gamma (\alpha^k)^{(x-a\gamma)k^{-1}} \bmod p \\ &= (\alpha^a)^\gamma (\alpha)^{(x-a\gamma)} \bmod p \\ &= \alpha^x \bmod p \end{aligned}$$

16.3 Bad usage

Revealing k or using the same k twice can cause forgery of chosen messages. If k is known, one can compute information on a from:

$$a\gamma \leftarrow x - \delta k \bmod p - 1.$$

If the same k is used for two messages, we obtain the following

$$\begin{aligned}\delta_1 &= k^{-1}(x_1 - a\gamma) \bmod p-1 \\ \delta_2 &= k^{-1}(x_2 - a\gamma) \bmod p-1\end{aligned}$$

Thus

$$(\delta_1 - \delta_2)k = x_1 - x_2 \bmod p-1.$$

If $\delta_1 - \delta_2 \neq 0 \bmod p-1$, we can compute

$$d \leftarrow gcd(\delta_1 - \delta_2, p-1).$$

Since $d|\delta_1 - \delta_2$ and $d|p-1$, we know that $d|(x_1 - x_2)$. Thus we can write

$$\begin{aligned}x' &:= \frac{x_1 - x_2}{d} \\ \delta' &:= \frac{\delta_2 - \delta_1}{d} \\ p' &:= \frac{p-1}{d}.\end{aligned}$$

The equation becomes

$$x' = k\delta' \bmod p'$$

Since $gcd(\delta', p') = 1$, we can compute $(\delta')^{-1}$, then

$$k = x'(\delta')^{-1} \bmod p'$$

This yields d candidate values for k, we can choose the right k by verifying with the signature verification function. From k we can then deduce a.

16.4 Forgeries

Some forgeries are now discussed by categories corresponding to the way *Oscar* forges a signature:

- Given x, set a γ and then try to find δ.
  The problem at hand would be to solve for δ given $\beta^\gamma\gamma^\delta = \alpha^x \bmod p$, which is equivalent to solving for δ given

  $$\gamma^\delta = (\alpha^x)(\beta^\gamma)^{-1} \bmod p$$

  this is equivalent to the DLP mod$p$.

- Given x, set a δ, try to find γ.
  This reduces to trying to find γ given

$$\beta^{\gamma}\gamma^{\delta} = \alpha^{x} \bmod p.$$

  No efficient solution to this problem is known, this problem is not known to be related to any "well-studied" problem like DLP.

- Given x, try to simultaneously find δ and γ.
  There is no known way of doing this.

Is it possible for *Oscar* to sign a random message? If *Oscar* chooses γ and δ and then tries to solve for x, he must compute $log_{\alpha}(\beta^{\gamma}\gamma^{\delta})$, yet another instance of the DLP.

However, there is a way for *Oscar* to sign a "random" message by choosing γ, δ and x simultaneously, it is described by the following algorithm

Algorithm 16.2 (Forge ElGamal)

1: *Pick i and j such that $0 \leq i, j \leq p-2$ and $gcd(j, p-1) = 1$.*

2: $\gamma \leftarrow \alpha^{i}\beta^{j} \bmod p$.

3: $\delta = -\gamma j^{-1} \bmod p-1$, $x \leftarrow -\gamma i j^{-1} \bmod p-1$.

Theorem 16.2 *The above algorithm gives a valid signature.*

Proof.

$$\begin{aligned}
\beta^{\gamma}\gamma^{\delta} &= \beta^{\gamma}(\alpha^{i}\beta^{j})^{-\gamma j^{-1}} \bmod p \\
&= \beta^{\gamma}(\alpha^{-\gamma i j^{-1}}\beta^{-\gamma}) \bmod p \\
&= \alpha^{-\gamma i j^{-1}} \bmod p \\
&= \alpha^{x} \bmod p
\end{aligned}$$

Note: in a variation of the ElGamal signature scheme, one uses $h(x)$ instead of x, where h is a cryptographic hash function. Other than the fact that this enables signatures of data of arbitrary size, it also prevents the above forgery from being successful.

It is also possible for $Oscar$ to forge some message given a previous message and signature (x, γ, δ).

Algorithm 16.3 (Forge From Previous ElGamal)

1: *Pick* h, i, j *such that* $0 \leq h, i, j \leq p-2$ *and* $gcd(h\gamma - j\delta, p-1) = 1$.

2: $\lambda \leftarrow \gamma^h \alpha^i \beta^j \bmod p$

3: $\mu \leftarrow \delta\lambda(h\gamma - j\delta)^{-1} \bmod p-1$

4: $x' = \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod p-1$.

Theorem 16.3 *The above algorithm gives* (x', λ, μ) *such that*

$$\beta^\lambda \lambda^\mu = \alpha^{x'} \bmod p$$

16.5 Digital Signature Standard

The Digital Signature Standard (DSS) describes a Digital Signature Algorithm (DSA) in FIPS 186, it is a variation of the ElGamal system. DSS relieves the burden of oversized signatures (with ElGamal signing a 160-bit message using a 512 bit prime, for example, produces a signature that is 1024 bits long, DSS would produce a 320-bit signature).

Algorithm 16.4 (DSA key generation)

1: *Choose a* 512-*bit prime* p

2: *Pick a* 160-*bit prime* q *such that* $q|p-1$.

3: *Choose* $\alpha \in Z_p^*$ *a* q^{th} *primitive root of* 1 *mod* p.

4: *Compute* $\beta \leftarrow \alpha^a \bmod p$.

5: **RETURN** *public* p, q, α, β *and private* a

Note: To pick α, you can start by picking α_o a primitive element of $\mathbb{Z}_p^*$ and then computing $\alpha \leftarrow \alpha_o^{(p-1)/q}$.

Algorithm 16.5 (DSS signature)

1: *Pick a random* k *such that* $1 \leq k \leq p-2$.

2: $\gamma \leftarrow (\alpha^k \bmod p) \bmod q$

3: $\delta \leftarrow (x + a\gamma)k^{-1} \bmod q$.

4: **IF** $\delta = 0$, **GOTO step 1**

5: **RETURN** $s = (\gamma, \delta)$.

Verification:

$$\begin{aligned} & e_1 \leftarrow x\delta^{-1} \bmod q \\ & e_2 \leftarrow \gamma\delta^{-1} \bmod q \\ Ver_K(x,\gamma,\delta) &= true \iff (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma. \end{aligned}$$

If the signature was constructed correctly, then the verification will succeed since

$$\begin{aligned} (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q &\equiv \alpha^{e_1}\alpha^{ae_2} \\ &\equiv \alpha^{x\delta^{-1}}\alpha^{a\gamma\delta^{-1}} \\ &\equiv \alpha^{\delta^{-1}(x+a\gamma)} \\ &\equiv \alpha^k \\ &= (\gamma \bmod p) \bmod q. \end{aligned}$$

16.6 Undeniable signatures

In this type of signature scheme, the verification protocol requires the co-operation of the signer. The scheme is composed of three components: a signing algorithm, a verification protocol and a disavowal protocol. A disavowal protocol enables one to determine whether the signer is attempting to disavow a valid signature or whether the signature was forged.

16.6.1 Chaum-Van Antwerpen's scheme

The first undeniable signature scheme was introduced in [**?**].

Algorithm 16.6 (Chaum-Van Antwerpen key generation)

1: *Select a random prime* $p = 2q + 1$, *where* q *is also prime.*

2: *Select* $\alpha \leftarrow y^2 \bmod p$, *for a random* $y \in_R \{2, 3, \ldots, p-2\}$.

3: *Select a random* $a \in_R \{1, 2, \ldots, q-1\}$, $\beta \leftarrow \alpha^a \bmod p$.

4: RETURN *public* (p, α, β) *and private* a.

α is selected in such a way as to be a generator of the subgroup of order q in $\mathbb{Z}_p^*$. The scheme operates in $\mathbb{Z}_p$, however, we need to be able to compute in a multiplicative subgroup of $\mathbb{Z}_p^*$. Picking $p = 2q + 1$, p, q primes, enables us to do this, and in a large as possible subgroup.

Algorithm 16.7 (Chaum-Van Antwerpen signature)

1: $s \leftarrow x^a \bmod p$.

2: RETURN s

Algorithm 16.8 (Chaum-Van Antwerpen verification)

1: *Bob selects random secret integers* $e_1, e_2 \in_R \{1, 2, \ldots, q-1\}$.

2: *Bob computes* $z \leftarrow s^{e_1}\beta^{e_2} \bmod p$ *and sends* z *to Alice.*

3: *Alice computes* $w = (z)^{a^{-1}} \bmod p$ *and sends* w *to Bob.*

4: *Bob accepts* $\iff$ $w = x^{e_1}\alpha^{e_2} \bmod p$

If *Alice* is honest, *Bob* will accept:

$$\begin{aligned} w &= (z)^{a^{-1}} \bmod p \\ &= (s^{e_1}\beta^{e_2})^{a^{-1}} \bmod p \\ &= (x^{ae_1}\alpha^{ae_2})^{a^{-1}} \bmod p \\ &= x^{e_1}\alpha^{e_2} \bmod p \end{aligned}$$

Theorem 16.4 *Suppose* $s \neq x^a \bmod p$ *is a forged signature, the probability* *that Bob will accept the signature in the above algorithm is* $1/q$.

The following disavowal protocol allows *Alice* to convince *Bob* that a certain value is not a valid signature. However, *Alice* might attempt to disavow a valid signature. The following protocol essentially performs the verification protocol twice and checks that *Alice* is not cheating:

Algorithm 16.9 (Chaum-Van Antwerpen Disavowal)

1: *Bob randomly selects* $e_1, e_2 \in_R \{1, 2, \ldots, q-1\}$

2: *Bob computes* $z \leftarrow s^{e_1}\beta^{e_2} \bmod p$*, sends* z *to Alice.*

3: *Alice computes* $w = (z)^{a^{-1}} \bmod p$ *and sends* w *to Bob.*

4: IF $w = x^{e_1}\alpha^{e_2} \bmod p$ **THEN RETURN valid**
/* *Bob concludes that Alice is trying to disavow a valid sig* */

5: *Bob selects random* $e_3, e_4 \in_R \{1, 2, \ldots, q-1\}$

6: *Bob computes* $z' \leftarrow s^{e_3}\beta^{e_4} \bmod p$*, sends* z' *to Alice.*

7: *Alice computes* $w' \leftarrow (z')^{a^{-1}} \bmod p$ *and sends* w' *to Bob.*

8: IF $w' = x^{e_3}\alpha^{e_4} \bmod p$ **THEN RETURN valid**
/* *Bob concludes that Alice is trying to disavow a valid sig* */

9: *Bob computes* $c \leftarrow (w\alpha^{-e_2})^{e_3} \bmod p$*,* $c' \leftarrow (w'\alpha^{-e_4})^{e_1} \bmod p$

10: IF $c = c'$ **THEN RETURN forgery**
/* *Bob concludes that the sig was a forgery* */

11: ELSE RETURN valid
/* *Bob concludes that Alice is trying to disavow a valid sig* */

Theorem 16.5 *The probability for Alice to successfully disavow a valid signature* $s = x^a \bmod p$*, in the above algorithm, is* $1/q$.

References

[BB88] Pierre Beauchemin and Gilles Brassard. A generalization of hellman's extension of shannon's approach to cryptography. In Carl Pomerance, editor, *Proc. CRYPTO 87*, pages 461–461. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 293.

[BS93] E. Biham and A. Shamir. *A Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[dB88] Bert den Boer. Cryptanalysis of F.E.A.L. In Christoph G. Günther, editor, *Advances in Cryptology—EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 293–299. Springer-Verlag, 25–27 May 1988.

[Hel80] M. E. Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT-26:401–406, 1980.

[Mat94] M. Matsui. Linear cryptanalysis method for DES cipher. *Lecture Notes in Computer Science*, 765:386–397, 1994.

[Mur90] S. Murphy. The cryptoanalysis of FEAL-4 with 20 chosen plaintexts. *Journal of Cryptology*, 2:145–154, 1990.

[MVV97] A. J. (Alfred J.) Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.

[NBS93] Data encryption standard (des). Technical Report FIPS PUB 46-2, National Bureau of Standards, 1993.

[NIS80] Des modes of operation. Technical Report FIPS PUB 81, National Institute of Standards and Technology, december 1980.

[NIS99] http://www.nist.gov/aes/, 1999.

[Sha48] C. E. Shannon. A mathematical theory of communication. volume 27, pages 379–423, 623–656, 1948.

[Sta99] William Stallings. *Cryptography and network security: principles and practice*. Prentice-Hall, Inc., Upper Saddle River, NJ 07458, USA, second edition, 1999.

[Sti95] Douglas R. Stinson. *Cryptography Theory and Practice*. CRC Press, Boca Raton, 1995.

[VK84] V. L. Voydock and S. T. Kent. Security mechanisms in a transport layer protocol. *Computer Networks*, 8:433–450, 1984.