8. DIGITAL SIGNATURES

Digital signatures are one of the most important inventions of modern cryptography.

The problem is how can a user (digitally) sign a (digital) message such that everybody (or the intended addressee only) can verify the digital signature and the signature is good enough also for legal purposes.

Moreover, each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behind of other users.

The main difference from a handwritten signature is that digital signature of a message is intimately connected with the message, and for different messages is different, whereas the handwritten signature is adjoined to the message and always look the same.

Technically, a digital signature is realized by a **signing algorithm** and it is verified by a **verification algorithm**.

A copy of digital (conventional) signature is identical (usually distinguishable) to (from) the origin. A care has therefore to be made that a classical signature is not misused.

This chapter contains an overview of the main techniques for performing and verifying digital signatures (as well as some attacks to them).

DIGITAL SIGNATURE of ONE BIT

Let us start with a simple illustrating (non-practical) example how to sign a single bit.

Design of a digital signature scheme

A one-way function $f(x)$ is chosen.

Two integers k_0 and k_1 are chosen, kept secret and

$$f, (0, s_0), (1, s_1)$$

are made public, where

$$s_0 = f(k_0), s_1 = f(k_1)$$

Signature of a bit b

$$(b, k_b).$$

Verification of the signature

$$s_b = f(k_b)$$

SECURITY?

RSA signatures and attacks on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d.

Signing of a message w

$$s = (w, \sigma), \sigma = w^d$$

Verification of a signature $s = (w, \sigma)$

$$w = \sigma^e.$$

Attacks

- It might happen that Bob accepts a signature not produced by Alice. Indeed, let Eve, using Alice's public key, computes w^e and says that (w^e, w) is a message signed by Alice.

  Everybody verifying Alice's signature gets $w^e = w^e$.

- Some new signatures can be produced without knowing secret key.

  Indeed, is σ_1 and σ_2 are signatures for w_1 and w_2, then $\sigma_1\sigma_2$ and σ_1^{-1} are signatures for w_1w_2 and w_1^{-1}.

ENCRYPTION versus SIGNATURE

Cryptosystem: Let each user U uses a cryptosystem with encryption and decryption algorithms: e_U, d_U

Message: w

PUBLIC-KEY CRYPTOGRAPHY

Encryption: $e_U(w)$

Decryption: $d_U(e_U(w))$

PUBLIC-KEY SIGNATURES

Signing: $d_U(w)$

Verification: $e_U(d_U(w))$

DIGITAL SIGNATURE SYSTEMS

A digital signature system (DSS) consists:

- $\mathcal{P}$ — The space of possible plaintexts (messages).
- $\mathcal{S}$ — The space of possible signatures.
- $\mathcal{K}$ — The space of possible keys.
- For each $k \in \mathcal{K}$ there is a signing algorithm $sig_k \in \mathcal{S}_a$ and a corresponding verification algorithm $ver_k \in \mathcal{V}$ such that
  - $sig_k : \mathcal{P} \to \mathcal{S}$.
  - $ver_k : \mathcal{P} \otimes \mathcal{S} \to \{true, false\}$

and

$$ver_k(w, s) = \begin{cases} \text{true}, & \text{if } s = sig(w); \\ \text{false}, & \text{otherwise.} \end{cases}$$

Functions sig_k and ver_k should be computable in polynomial time.

Verification algorithm can be publically known; signing algorithm (actually only its key) should

be kept secret.

FROM PKC to DSS

Any public-key cryptosystem in which the plaintext and cryptotext space are the same can be used for digital signature.

Signing of a message w by a user A so that any user can verify the signature:

$$d_A(w).$$

Signing of a message w by a user A so that only user B can verify the signature;

$$e_B(d_A(w)).$$

Sending a message w and a signed message digest of w obtained by using a (standard) hash function h:

$$(w, d_A(h(w))).$$

If only signature (but not the encryption of the message) are of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w)).$$

Caution: Signing a message w by A for B by

$$e_B(d_A(w))$$

is O.K., but the symmetric solution with encoding was

$$c = d_A(e_B(w))$$

is not good. Indeed, an active enemy, the <u>tamperer</u>, can intercept the message, then compute

$$d_T(e_A(c)) = d_T(e_B(w))$$

and send it to Bob, pretending it is from him (without being able to decrypt the message).

ElGamal signatures

Choose: prime p, integers $1 \leq q \leq x \leq p$, q be a primitive element of Z_p^*;
Compute: $y = q^x \bmod p$

$$\mathbf{key}\ \mathcal{K} = (p, q, x, y)$$
$$\mathbf{public\ key}\ (p, q, y) \text{ – } \mathbf{trapdoor}\text{: } x$$

Signature of a message w: Let $r \in \mathbf{Z}_{p-1}^*$ be randomly chosen and kept secret.

$$sig(w, r) = (a, b),$$

where

$$a = q^r \bmod p$$

and

$$b = (w - xa)r^{-1} \pmod{p-1}$$

that is b should be a solution of the equation $w \equiv rb + xa \pmod{p-1}$.

Verification:

$$ver_{\mathcal{K}}(w, a, b) = \mathbf{true} \Leftrightarrow y^a a^b \equiv q^w (\bmod\, p)$$

(Indeed: $y^a a^b \equiv q^{ax} q^{rb} \equiv q^{ax+w-ax+k(p-1)} \equiv q^w (\bmod\, p)$)

Example choose: $p = 11, q = 2, x = 8,$
compute: $y = 2^8 \bmod 11 = 3.$
Let $w = 5$. Choose $r = 9$. – O.K. because $\gcd(9, 10) = 1$.
compute $a = 2^9 \bmod 11 = 6,$
, solve equation: $5 \equiv 8 \cdot 6 + 9b \pmod{10}$,
$7 \equiv 9b \pmod{10} \Rightarrow b = 3,$
signature: $(6, 3)$.

Security of ElGamal signatures

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x- secret; p, q and $y = q^x \bmod p$ - public):

$$\mathit{sig}(w, r) = (a, b);$$

where r is random and

$$a = q^r \bmod p; \quad b = (w - xa)r^{-1} \bmod p - 1.$$

1. First suppose that Eve tries to forge signature for a new message w without knowing x.
   - If Eve first chooses a value a and tries to find the corresponding b, it has to compute the discrete logarithm

     $$\lg_a q^w y^{-a},$$

     because $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$.
   - If Eve first chooses b and then tries to find a, she has to solve the equation

     $$y^a a^b \equiv q^{xa} q^{rb} \equiv q^w \pmod p.$$

     It is not known whether this equation can be solved for a efficiently.
2. If Eve chooses a and b and tries to determine w, then she has to compute discrete logarithm

   $$\lg_q y^a a^b.$$

   Hence, Eve cannot sign a "random" message this way.

Forging and misusing of ElGamal signatures

There are ways how to produce, using ElGamal signature scheme, valid but forged signatures, However, they do not allow an opponent to forge signatures on messages of his/her choice.

For example, if $0 \leq i, j \leq p-2$ and $gcd(j, p-1) = 1$, then for

$$a = q^i y^j \bmod p; b = -aj^{-1} \bmod (p-1); w = -aij^{-1} \bmod (p-1)$$

the pair

(a, b) is a valid signature of the message w.

This can be easily shown by checking the verification condition.

There are several ways ElGamal signatures can be broken if they are used not carefully enough.

For example, the random r used in the signature should be kept secret. Otherwise the system can be broken and signatures forged. Indeed, if r is known, then x can be computed by

$$x = (w - rb)a^{-1} \bmod (p-1)$$

and once x is known Eve can forge signatures at will.

Another misuse of the ElGamal signature system is to use the same r to sign two messages. In such a case x can be computed and system can be broken.

Digital Signature Standard

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

Design of DSA

1. Global public key components are chosen:
- p – a random l-bit prime, $512 \leq l \leq 1024$, is chosen, $l = 64k$.
- q – a random 160-bit prime dividing $p - 1$ is chosen.
- $r = h^{\frac{p-1}{q}} \bmod p$ is computed, where h is a random primitive element of $\mathbf{Z}_p$, such that $r \neq 1$ (r is a qth root of $1 \bmod p$).

2. User's private key is chosen:
- x – a random integer, $0 < x < q$, is chosen and

$$y = r^x \bmod p$$

is computed.

3. Key is $\mathcal{K} = (p, q, r, x, y)$:

Signing and Verification

Signing of a 160-bit plaintext w

- choose random $0 < k < q$, gcd(k,q)=1
- compute $a = (r^k \bmod p) \bmod q$
- compute $b = k^{-1}(w + xa) \bmod q$ where $kk^{-1} \equiv 1 (\bmod q)$
- signature: *sig*$(w, k) = (a, b)$

Verification – signature (a, b)
- compute $z = b^{-1} \bmod q$
- compute $u_1 = wz \bmod q$, $u_2 = az \bmod q$
- verification:

ver$_{\mathcal{K}}(w, a, b) = \mathbf{true}$

$$\Leftrightarrow$$

$(r^{u_1} y^{u_2} \bmod p) \bmod q = a$

From ElGamal to DSA

DSA is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny.

Encryption of a message is usually done only once and therefore it usually suffices to use a cryptosystem that is secure **at the time of encryption**.

On the other hand, a signed message could be a contract or a will and it can happen that it will be needed to verify a signature **many years after the message is signed**.

Since ElGamal signature is no more secure than discrete logarithm, it is necessary to use large p, with at least 512 bits.

However, with ElGamal this would lead to signatures with at least 1024 bits what is too much for such applications as smart cards.

In DSA a 160 bit message is signed using 320-bit signature, but computation is done modulo with 512-1024 bits.

Fiat-Shamir signature scheme

Choose primes p, q, compute $n = pq$, choose security parameters t, k and choose:
as **publ. key** $v_1, \ldots, v_k$ and compute **secr. key** $s_1, \ldots, s_k,\ s_i = \sqrt{v_i^{-1}} \bmod n$.

Protocol for Alice to sign a message w and for Bob to verify her signature:
(1) Alice chooses t random integers $1 \leq r_1, \ldots, r_t < n$, computes $x_i = r_i^2 \bmod n,\ 1 \leq i \leq t$.
(2) Alice uses a publically known hash function h to compute

$$h(wx_1x_2 \ldots x_t)$$

and uses in the next its first $k \times t$ bits denoted as $b_{ij}, 1 \leq i \leq t,\ 1 \leq j \leq k$.
(3) Alice computes for $1 \leq i \leq t$

$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \bmod n$$

(4) Alice sends to Bob w, all b_{ij} and all y_i $\{b_{ij}$ and y_i form the signature$\}$
$\{$Bob already knows Alice's public key $v_1, \ldots, v_k\}$ and also h
(5) Bob computes for $1 \leq i \leq t$

$$z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \bmod n \ \{= r_i^2 \prod_{j=1}^{k} (v_j^{-1})^{b_{ij}} \prod_{j=1}^{k} v_j^{b_{ij}} = r_i^2 = x_i\}$$

and verifies that the first $k \times t$ bits of $h(wz_1 \ldots z_t)$ are the b_{ij} values that Alice sent to him.

Security of this signature scheme is 2^{-kt}.

Advantage over the RSA-based signature scheme. Only about 5% of modular multiplications is needed.

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a **subliminal channel**, a covert communications channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Ong-Schnorr-Shamir subliminal channel scheme

Alice and Bob create the following communication scheme:
They choose large n and an integer k such that $\gcd(n, k) = 1$.
They calculate h such that $h = k^{-2} \bmod n = (k^{-1})^2 \bmod n$.

Public key: h, n
Trapdoor information: k

Let secret message Alice wants to send be: w (it has to be such that $\gcd(w, n) = 1$)
Denote harmless message she uses by: w' (it has to be such that $gcd(w', n) = 1$)

Signing by Alice:

$$S_1 = \frac{1}{2} \cdot \left(\frac{w'}{w} + w \right) \bmod n$$

$$S_2 = \frac{k}{2} \cdot \left(\frac{w'}{w} - w \right) \bmod n$$

Signature: (S_1, S_2) Alice then sends (w', S_1, S_2)

Signature verification by Walter: $w' \equiv S_1^2 - hS_2^2 \pmod{n}$

Decryption by Bob:

$$w = \frac{w'}{(S_1 + k^{-1}S_2)} \bmod n.$$

One-time signatures

Lamport signature scheme shows how to construct a signature scheme for one use from any one-way function.

Let k be a positive integer and let $\mathcal{P} = \{0,1\}^k$ be the set of messages.

Let $f : Y \to Z$ be a one-way function and let Y be the set of "signatures".

For $1 \leq i \leq k, j = 0, 1$ let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The key $\mathcal{K}$ consists of $2k$ y's and z's.
y's are secret, z's are public.

Signing of a message $x = x_1 \dots x_k \in \{0,1\}^k$

$$\textsf{\textit{sig}}(x_1 \dots x_k) = (y_{1,x_1}, \dots, y_{k,x_k}) = (a_1, \dots, a_k)$$

and

$$\textsf{\textit{ver}}_{\mathcal{K}}(x_1 \dots x_k, a_1, \dots, a_k) = \textbf{true} \Leftrightarrow f(a_i) = z_{i,x_i}, 1 \leq i \leq k$$

Eve cannot forge a signature because she is unable to invert one-way functions.

Important note: Lamport signature scheme can be used to sign only one message.

Undeniable and unclonable signatures

Undeniable signatures consist:

- Signing algorithm
- Verification protocol, that is a *challenge-and-response protocol.*

  In this case it is required that a signature cannot be verified without a cooperation of the signer (Bob).

  This protects Bob against the possibility that documents signed by him are duplicated and distributed without his approval.
- Disavowal protocol, by which Bob can prove that a signature is a forgery.

  This is to prevent Bob from disavowing a signature he made at an earlier time.

Chaum-van Antwerpen undeniable signature schemes (CAUSS)

- p, r are primes $p = 2r + 1$;
- $q \in \mathbf{Z}_p^*$ is of order r;
- $1 \leq x \leq r - 1, \quad y = q^x \bmod p$;
- G is a multiplicative subgroup of $\mathbf{Z}_p^*$ of order q (G consists of quadratic residues modulo p).

Key space: $\mathcal{K} = \{(p, q, x, y\}$; p, q, y are public, $x \in G$ is secret.

Signature: $s = sig_{\mathcal{K}}(w) = w^x \bmod p$.

Fooling and Disallowed protocol

Since it holds:

Theorem If $s \neq w^x \bmod p$, then Alice will accept s as a valid signature for w with probability $\frac{1}{r}$.

Bob cannot fool Alice except with very small probability and security is unconditional (that is, it does not depend on any computational assumption).

Disallowed protocol

Basic idea: After receiving a signature s Alice initiates two independent and unsuccessful runs of the verification protocol. Finally, she performs a "consistency check" to determine whether Bob has formed his responses according to the protocol.

- Alice $e_1, e_2 \in \mathbf{Z}_r^*$.
- Alice computes $c = s^{e_1} y^{e_2} \bmod p$ and sends it to Bob.
- Bob computes $d = c^{x^{-1} \bmod r} \bmod p$ and sends it to Alice.
- Alice verifies that $d \not\equiv w^{e_1} q^{e_2} \pmod p$.
- Alice $f_1, f_2 \in \mathbf{Z}_r^*$.
- Alice computes $C = s^{f_1} y^{f_2} \bmod p$ and sends it to Bob.
- Bob computes $D = C^{x^{-1} \bmod r} \bmod p$ and sends it to Alice.
- Alice verifies that $D \not\equiv w^{f_1} q^{f_2} \pmod p$.
- Alice concludes that s is a forgery iff

$$(dq^{-e_2})^{f_1} \equiv (Dq^{-f_2})^{e_1} \pmod p.$$

CONCLUSIONS

It can be shown:

- Bob can convince Alice that an invalid signature is a forgery. In order to that it is sufficient to show that if $s \not\equiv w^x$, then

$$(dq^{-e_2})^{f_1} \equiv (Dq^{-f_2})^{e_1} \pmod{p}$$

  what can be done using congruency relation from the design of the signature system and from the disallowed protocol.

- Bob cannot make Alice believe that a valid signature is a forgery, except with a very small probability.

Signing of fingerprints

Signatures scheme presented so far allow to sign only "short" messages. For example, DSS is used to sign 160 bit messages (with 320-bit signatures.

A naive solution is to break long message into a sequence of short ones and to sign each block separately.

Disadvantages: signing is slow and for long signatures integrity is not protected.

A better solution is to use fast public hash functions h which maps a message of any length to a fixed length fingerprint. The fingerprint is then signed.

Scheme:

message	w	arbitrary length
message digest	$z = h(w)$	160bits
signature	$y = \mathrm{sig}(z)$	320bits

If Bob wants to sign a message w he sends $(w, \mathrm{sig}(h(w)))$.

Collision-free hash functions revisited

For a hash function it is necessary to be good enough for creating fingerprints that do not allow various forgeries of signatures.

For example, Eve can start with a valid signature $(w, \textit{sign}(h(w)))$, compute $h(w)$ and try to find w' such that $h(w) = h(w')$. If she succeeds, then

$$(w', \textit{sign}(h(w)))$$

is a valid signature of a message, but a **forgery**.

In order to prevent the above type of attacks, and some other, it is required that a hash function h satisfies the following **collision-free property.**

Definition A hash function h is (strongly) collision-free if it is computationally infeasible to find messages w and w' such that $h(w) = h(w')$.

Another possible attack: Eve computes a signature y of a random fingerprint z and then find an x such that $z = h(x)$. In such a case (x, y) is a valid signature.

In order to prevent the above attack, it is required that in signatures we use **one-way hash functions**.

It is not difficult to show that for hash-functions (strong) collision property implies the one-way property.

Theorem Let $h : X \to Z$ be a hash function where $|X|$ and $|Z|$ are finite and $|X| \geq 2|Z|$. Suppose that A is an inversion algorithm for h. Then there exists a Las Vegas algorithm to find a collision for h with probability at least $\frac{1}{2}$.

Timestamping

There are various way that a signing algorithm may be compromised.

If Eve determines the secret key of Bob, then she can forge signatures on any message she likes. If this happens, then authenticity of all messages signed by Bob before Eve got the secret key is to be questioned.

The key problem is that there is no way to determine when a message was signed.

A **timestamping** should provide proof that a message was signed at a certain time.

A method for timestamping of signatures:

In the following *pub* denotes some publically known information that could not be predicted before the day of the signature (for example, stock-market data).

Timestamping by Bob of a signature on a message w.

- Bob computes $z = h(w)$;
- Bob computes $z' = h(z||\textit{pub})$.
- Bob computes $y = \textit{sig}(z')$.
- Bob publishes $(z, \textit{pub}, y)$ in the next days's newspaper.

It is now clear that signature could not be done after triple $(x, \textit{pub}, y)$ was published, but also not before date *pub* was known.

Digital signature with encryption + resending the message as a receipt – a possible attack

Digital signature + encryption

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$ and sends it to Bob.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

Resending the message as a receipt

5. Bob signs and encrypts the message sending $e_A(s_B(w))$.
6. Alice decrypts the message and verifies the signature.

Assume now: $v_x = e_x, s_x = d_x$ for all users x.

Attack:
1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him.
3. Bob decrypts and verifies the message by computing

$$e_M(d_B(c_B(d_A(w)))) = e_M(d_A(w)) \qquad \text{– \{a garbage\}.}$$

4. Bob goes on with the protocol and sends Mallot the receipt:

$$e_M(d_B(e_M(d_A(w))))$$

5. Mallot gets w. Indeed, $e_A(d_M(e_B(d_M(e_M(d_B(e_M(d_A(w)))))))) = w$.

A MAN-IN-THE-MIDDLE-ATTACK

Consider the following protocol:

1. Alice sends Bob the pair $(e_B(e_B(w)A), B)$ to B.
2. Bob uses d_B to get A and w, and acknowledges by sending the pair $(e_A(e_A(w)B), A)$ to Alice.

(Here the function e and d are assumed to operate on numbers, names $A, B, \ldots$ are sequences of digits and $e_B(w)A$ is a sequence of digitals obtained by concatenating $e_B(w)$ and A.)

What can an active eavesdropper C do?

- C can learn $(e_A(e_A(w)B), A)$ and therefore $e_A(w'), w' = e_A(w)B$.
- C can now send to Alice the pair $(e_A(e_A(w')C), A)$.
- Alice, thinking that this is the step 1 of the protocol, acknowledges by sending the pair $(e_C(e_C(w')A), C)$ to C.
- C is now able to learn w' and therefore also $e_A(w)$.
- C now sends to Alice the pair $(e_A(e_A(w)C), A)$.
- Alice acknowledges by sending the pair $(e_C(e_C(w)A), C)$.
- C is now able to learn w.

Probabilistic signature schemes (PSS)

Let us have a a trapdoor permutation

$$f : D \to D, D \subset \{0,1\}^n,$$

a pseudorandom bit generator

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, w \to (G_1(w), G_2(w))$$

and a hash function

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w||r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m||(G_1(m) \oplus r)||G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

Verification of a signed message (w, σ).

1. Compute $f(\sigma)$ and decompose $f(\sigma) = m||t||u$, where $|m| = l$, $|t| = k$ and $|u| = n - (k + l)$.
2. Compute $r = t \oplus G_1(m)$.
3. Accept signature σ if $h(w||r) = m$ and $G_2(m) = u$; otherwise reject it.

Authenticated Diffie-Hellman key exchange

Let each user U have a signature function s_U and a verification algorithm v_U.

The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and avoids the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in \mathbf{Z}_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \bmod p$, and Bob computes $q^y \bmod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \pmod{p}$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \pmod{p}$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.
9. Alice verifies, using an authority, that v_B is Bob's verification algorithm.
10. Alice uses v_B to verify Bob's signature.
11. Alice sends $e_K(s_A(q^x, q^y))$ to Bob.
12. Bob decrypts, verifies v_A, and verifies Alice's signature.

An enhanced version of the above protocol is known as Station-to-Station protocol.

Security of digital signature

It is again very non-trivial to define security of digital signature.

Definition A chosen message attack is a process which on input of a verification key can obtain signature (corresponding to the given key) to message of its choice.

A chosen message attack is considered to be successful (in so called existential forgery) if it outputs a valid signature for a message for which it has not requested a signature during the attack.

A signature scheme is secure (or unforgeable) if every feasible chosen message attack succeeds with at most negligible probability.