Part VII

Digital signatures

CHAPTER 7: DIGITAL SIGNATURES

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

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

Example: Assume that each user A uses a public-key cryptosystem (e_A,d_A).

A way to sign a message w by a user A, so that any user can verify the signature:

$$d_A(w)$$

A way to sign a message w by a user A so that only user B can verify the signature:

$$e_B(d_A(w))$$

Example Assume Alice succeeds to factor the integer Bob used, as modulus, to sign his will, using RSA, 20 years ago. Even if the key has already expired, Alice can rewrite Bob's will, leaving fortune to her, and date it 20 years ago.

Moral: It may pay off to factor a single integers using many years of many computers power.

DIGITAL SIGNATURES - BASIC GOALS

Digital signatures should be such that 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 behalf of other users.

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

Technically, a digital signature signing is performed by a signing algorithm and a digital signature is verified by a verification algorithm.

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

This chapter contains some of the main techniques for design and verification of digital signatures (as well as some possible attacks on them).

DIGITAL SIGNATURES - OBSERVATION

Can we make digital signatures by digitalizing our usual signature and attaching them to the messages (documents) that need to be signed?

No, because such signatures could be easily removed and attached to some other documents or messages.

Key observation: Digital signatures have to depend not only on the signer, but also on the message that is being signed.

A SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- P - the space of possible plaintexts (messages).
- S - the space of possible signatures.
- K - the space of possible keys.
- For each $k \in K$ there is a signing algorithm sig_k and a corresponding verification algorithm ver_k such that

$$sig_k : P \rightarrow S.$$

$$ver_k : P \otimes S \rightarrow \{true, false\}$$

and

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

Algorithms sig_k and ver_k should be computable in polynomial time.

Verification algorithm can be publicly known; signing algorithm (actually only its key) should be kept secret

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication and non-repudiation of messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A Digital Signature Scheme (M, S, K_s, K_v) is given by:

- M - a set of messages to be signed
- S - a set of possible signatures
- K_s - a set of private keys for signing
- K_v - a set of public keys for verification

Moreover, it is required that:

- For each k from K_s, there exists a single and easy to compute signing mapping

$$sig_k\colon \{0,1\}^* \times M \rightarrow S$$

- For each k from K_v there exists a single and easy to compute verification mapping

$$ver_k\colon M \times S \rightarrow \{true, false\}$$

such that the following two conditions are satisfied:

DIGITAL SIGNATURES SCHEMES II

Correctness:

For each message m from M and public key k in K_v, it holds

$$ver_k(m, s) = \text{true}$$

if there is an r from $\{0,1\}^*$ such that

$$s = sig_l(r, m)$$

for a private key l from K_s corresponding to the public key k.

Security:

For any w from M and k in K_v , it is computationally infeasible, without the knowledge of the private key corresponding to k, to find a signature s from S such that

$$ver_k(w, s) = \text{true}.$$

A COMMENT ON DIGITAL SIGNATURE SCHEMES

Sometimes it is said that a digital signature scheme contains also a key generation algorithm that selects uniformly and randomly a secret key (from a set of potential secret keys) and outputs this secret key and the corresponding private key.

ATTACK MODELS on DIGITAL SIGNATURES

Basic attack models

KEY-ONLY ATTACK : The attacker is only given the public verification key.

KNOWN SIGNATURES ATTACK : The attacker is given valid signatures for several messages known but not chosen by the attacker.

CHOSEN SIGNATURES ATTACK : The attacker is given valid signatures for sever al messages chosen by the attacker.

BASIC ATTACKS on DIGITAL SIGNATURES

Total break of a signature scheme: The adversary manages to recover the secret key from the public key.

Universal forgery: The adversary can derive from the public key an algorithm which allows to forge the signature of any message.

Selective forgery: The adversary can derive from the public key a method to forge signatures of selected messages (where selection was made prior the knowledge of the public key).

Existential forgery: The adversary is able to create from the public key a valid signature of a message m (but has no control for which m).

A DIGITAL SIGNATURE of one BIT

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

Design of the signature scheme:

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

Two integers k_0 and k_1 are chosen and kept secret by the signer, and three items

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

are made public, where

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

Signature of a bit b:

$$(b,\ k_b).$$

Verification of such a 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 and modulus n.

Signing of a message w:

$$s = (w, \sigma),\text{ where } \sigma = w^d \bmod n$$

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

$$w = \sigma^e \bmod n?$$

Attacks

- It might happen that Bob accepts a signature not produced by Alice. Indeed, let Eve, using Alice's public key, compute w^e and say 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 the 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}.

ENCRYPTIONS versus SIGNATURES

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

Let w be a message

PUBLIC-KEY ENCRYPTIONS

Encryption:	$e_U(w)$
Decryption:	$d_U\ (e_U(w))$

PUBLIC-KEY SIGNATURES

Signing:	$d_U(w)$
Verification of the signature:	$e_U\ (d_U(w))$

FROM PKC to DSS - again

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

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

ElGamal SIGNATURES

Design of the ElGamal digital signature system: choose: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\text{Compute: } y = q^x \mod p$$

key K = (p, q, x, y)

public key (p, q, y) - trapdoor: x

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

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

$$\text{where } a = q^r \mod p$$

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

Verification: accept a signature (a,b) of w as valid if

$$y^a a^b \equiv q^w \pmod p$$

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

ElGamal SIGNATURE - EXAMPLE

Example

choose: p = 11, q = 2, x = 8

compute: $y = 2^8$ mod 11 = 3

w = 5 is signed as (a,b), where $a = q^r \mod p, w = xa + rb \bmod (p-1)$

choose r = 9 – (this choice is O.K. because gcd(9, 10) = 1)

compute a = 2^9 mod 11 = 6

solve equation: $5 \equiv 8 \cdot 6 + 9b \pmod{10}$

that is $7 \equiv 9b \pmod{10} \Rightarrow$ 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 mod p - public):

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

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

1. First suppose 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 $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$) what is infeasible.
   - 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 any given b efficiently.
2. If Eve chooses a and b and tries to determine such w that (a,b) is signature of w, then she has to compute discrete logarithm
   $$lg_q y^a a^b.$$
   Hence, Eve can not sign a "random" message this way.

FORGING and MISUSING of ElGamal SIGNATURES

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

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

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

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 not used 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

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

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 the system can be broken.

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA, accepted in 1994, 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. Why?

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 the 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.

DIGITAL SIGNATURE STANDARD I

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. The following global public key components are chosen:
   - p - a random l-bit prime, $512 \le l \le 1024$, l = 64k.
   - q - a random 160-bit prime dividing p -1.
   - r = $h^{(p-1)/q}$ mod p, where h is a random primitive element of Z_p, such that $r > 1$, $r \neq 1$ (observe that r is a q-th root of 1 mod p).
2. The following user's private key component is chosen:
   - x - a random integer (once), $0 < x < q$,
3. The following value is also made public
   - y = r^x mod p.
4. Key is K = (p, q, r, x, y)

DIGITAL SIGNATURE STANDARD II

Signing and Verification

Signing of a 160-bit plaintext w

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

Verification of signature (a, b)

- compute $z = b^{-1} \mod q$
- compute $u_1 = wz \mod q$, $u_2 = az \mod q$

verification:

$$ver_K(w, a, b) = true \Leftrightarrow (r^{u_1} y^{u_2} \mod p) \mod q = a$$

From ElGamal to DSA - II

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

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.

Observe that y and a are also q-roots of 1. Hence any exponents of r,y and a can be reduced modulo q without affecting the verification condition.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \dots, v_k$ and compute, as a secret key, $s_1, \dots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice to sign a message w:

1. Alice chooses (as a security parameter) an integer t, t random integers $1 \leq r_1, \dots, r_t < n$, and computes $x_i = r_i^2 \mod n, 1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \dots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \dots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h {Bob already knows Alice's public key $v_1, \dots, v_k$}
5. Bob computes $z_1, \dots, z_k$
$$Z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \mod n = r_i^2 \prod_{j=1}^{k} (v_j^{-1})^{b_{ij}} \prod_{j=1}^{k} v_j^{b_{ij}} = r_i^2 = x_i$$
and verifies that the first $k \times t$ bits of $h(wx_1x_2 \dots x_t)$ are the b_{ij} values that Alice has sent to him.

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

   Advantage over the RSA-based signature scheme: only about 5% of modular multiplications are needed.

SAD STORY

Alice and Bob got to jail - and, unfortunately, to different jails.

Walter, the warden, allows them to communicate by network, but he will not allow their messages to be encrypted.

Problem: Can Alice and Bob set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves that they exchange contain no secret information?

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

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 communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:
They choose a large n and an integer k such that gcd(n, k) = 1.
They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n.$
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 a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)
Signing by Alice:

$$S_1 = \frac{1}{2} \cdot \left(\frac{w'}{w} + w\right) \mod n$$
$$S_2 = \frac{k}{2} \cdot \left(\frac{w'}{w} - w\right) \mod n$$

Signature: (S_1, S_2). Alice then sends to Bob (w', S_1, S_2)
Signature verification method for Walter: w' $= S_1^2 - hS_2^2(\mod n)$

Decryption by Bob: $w = \dfrac{w'}{(S_1 + k^{-1}S_2)} \mod n$

ONE-TIME SIGNATURES

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

Let k be a positive integer and let $P = \{0, 1\}^k$ be the set of messages.
Let f: $Y \rightarrow Z$ be a one-way function where Y is a 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 K consists of 2k y's and z's. y's are secret, z's are public.

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

$$sig(x_1 \ldots x_k) = (y_{1,x1}, \ldots, y_{k,xk}) = (a_1, \ldots, a_k) \text{ - notation}$$

and

$$ver_K(x_1 \ldots x_k, a_1, \ldots, a_k) = true \Leftrightarrow f(a_i) = z_{i,xi}, 1 \leq i \leq k$$

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.

SIGNING of FINGERPRINTS

Signature schemes 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.

The solution is to use a fast public **hash function h** which maps a message of any length to a fixed length hash. The hash is then signed.

Example:

message	w	arbitrary length
message digest	z = h (w)	160bits
El Gamal signature	y = sig(z)	320bits

If Bob wants to send a signed message w he sends (w, sig(h(w)).

TIMESTAMPING

There are various ways that a digital signature can be compromised.

For example: if Eve determines the secret key of Bob, then she can forge signatures of any Bob's message she likes. If this happens, 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 protocol should provide a proof that a message was signed at a certain time.

In the following pub denotes some publicly 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, using a hash function h.

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

It is now clear that signature could not be done after the triple (z, pub, y) was published, but also not before the date pub was known.

BLIND SIGNATURES

The basic idea is that Sender makes Signer to sign a message m without Signer knowing m, therefore blindly – this is needed in e-commerce.

Blind signing can be realized by a two party protocol, between the Sender and the Signer, that has the following properties.

- In order to sign (by a Signer) a message m, the Sender creates, using a blinding procedure, from the message m a new message $m*$ from which m can not be obtained without knowing a secret, and sends $m*$ to the Signer.
- The Signer signs the message $m*$ to get a signature s_{m^*} (of $m*$) and sends s_{m*} to the Sender. The signing is to be done in such a way that the Sender can afterwards compute, using an unblinding procedure, from Signer's signature s_{m*} of $m*$ – the signer signature s_m of m.

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Bob's RSA public key is (n, e) and his private key is d.
Let m be a message, $0 < m < n$,

PROTOCOL:

- Alice chooses a random $0 < k < n$ with $\gcd(n, k) = 1$.
- Alice computes $m^* = mk^e$ (**mod** n) and sends it to Bob (this way Alice blinds the message m).
- Bob computed $s^* = (m^*)^d$ (**mod** n) and sends s* to Alice (this way Bob signs the blinded message m*).
- Alice computes $s = k^{-1}s^*$(**mod** n) to obtain Bob's signature m^d of m (Alice performs unblinding of $m*$).

Verification is equivalent to that of the RSA signature scheme.

FAIL-THEN-STOP SIGNATURES

They are signatures schemes that use a trusted authority and provide ways to prove, if it is the case, that a powerful enough adversary is around who could break the signature scheme and therefore its use should be stopped.

The scheme is maintained by a trusted authority that chooses a secret key for each signer, keeps them secret, even from the signers themselves, and announces only the related public keys.

An important idea is that signing and verification algorithms are enhanced by a so-called proof-of-forgery algorithm. When the signer sees a forged signature he is able to compute his secret key and by submitting it to the trusted authority to prove the existence of a forgery and this way to achieve that any further use of the signature scheme is stopped.

So called Heyst-Pedersen Scheme is an example of a Fail-Then-Stop signature Scheme.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$.
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 and sends to Alice $e_A(s_B(w))$.
6. Alice decrypts the message and verifies the signature.

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

A SURPRISING ATTACK to PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).
3. Bob decrypts and "verifies" the message by computing
$$e_M(d_B(e_B(d_A(w)))) = e_M(d_A(w)) \text{ – a garbage.}$$
4. Bob goes on with the protocol and returns to Mallot the receipt:
$$e_M(d_B(e_M(d_A(w))))$$
5. Mallot can then get w.
Indeed, Mallot can compute
$e_A(d_M(e_B(d_M(e_M(d_B(e_M(d_A(w)))))))) =$ 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 strings and identificators $A, B, \ldots$ are strings.

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 SIGNATURES SCHEMES - PSS

Let us have integers k, l, n such that $k + l < n$, a 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).

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

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p - 1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p - 1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

An eavesdropper seems to need, in order to determine x from **X, q, p** and y from **Y, q, p**, a capability to compute discrete logarithms, or to compute q^{xy} from q^x and q^y, what is believed to be infeasible.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U has a signature algorithm 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 to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
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.

THRESHOLD DIGITAL SIGNATURES

The idea of a (t+1, n) threshold signature scheme is to distribute the power of the signing operation to (t+1) parties out of n.

A (t+1) threshold signature scheme should satisfy two conditions.

Unforgeability means that even if an adversary corrupts t parties, he still cannot generate a valid signature.

Robustness means that corrupted parties cannot prevent uncorrupted parties to generate signatures.

Shoup (2000) presented an efficient, non-interactive, robust and unforgeable threshold RSA signature schemes.

HISTORY of DIGITAL SIGNATURES

- In 1976 Diffie and Hellman were first to describe the idea of a digital signature scheme. However, they only conjectured that such schemes may exist.
- In 1977 RSA was invented that could be used to produce a primitive (not secure enough) digital signatures.
- The first widely marketed software package to offer digital signature was Lotus Notes 1.0, based on RSA and released in 1989
- ElGamal diital signatures were invented in 1984.
- In 1988 Goldwasser, Micali and Rivest were first to rigorously define (perfect0 security of digital signature schemes.

APPENDIX to CHAPTER 7

SPECIAL TYPES of DIGITAL SIGNATURES

- Append-Only Signatures (AOS) have the property that any party given an AOS signature $sig[M_1]$ on message M_1 can compute $sig[M_1 \| M_2]$ for any message M_2. (Such signatures are of importance in network applications, where users need to delegate their shares of resources to other users).
- Identity-Based signatures (IBS) at which the identity of the signer (i.e. her email address) plays the role of her public key. (Such schemes assume the existence of a TA holding a master public-private key pair used to assign secret keys to users based on their identity.)
- Hierarchically Identity-Based Signatures are such IBS in which users are arranged in a hierarchy and a user at any level at the hierarchy can delegate secret keys to her descendants based on their identities and her own secret keys.

GROUP SIGNATURES

- At Group Signatures (GS) a group member can compute a signature that reveals nothing about the signer's identity, except that he is a member of the group. On the other hand, the group manager can always reveal the identity of the signer.
- Hierarchical Group Signatures (HGS) are a generalization of GS that allow multiple group managers to be organized in a tree with the signers as leaves. When verifying a signature, a group manager only learns to which of its subtrees, if any, the signer belongs.

UNCONDITIONALLY SECURE DIGITAL SIGNATURES

Any of the digital signature schemes introduced so far can be forged by anyone having enough computer power.

Chaum and Roijakkers (2001) developed, for any fixed set of users, an unconditionally secure signature scheme with the following properties:

- Any participant can convince (except with exponentially small probability) any other participant that his signature is valid.
- A convinced participant can convince any other participant of the signature's validity, without interaction with the original signer.

BIRTHDAY PARADOX ATTACK on DIGITAL SIGNATURE

Assume Alice uses a hash function that produces 50 bits.

Fred, who wants Alice to sign a fraudulent contract, find 30 places in a good document, where he can make change in the document (adding a coma, space, ...) such that Alice would not notice that. By choosing at each place whether to make or not a change, he can produce 2^{30} documents essentially identical with the original good document.

Similarly, Fred makes 2^{30} changes of the fraudulent document.

Considering birthday problem with $n = 2^{50}$, $r = 2^{30}$ we get that $r = \sqrt{\lambda n}$, with $\lambda = 2^{10}$ and therefore with probability $1 - e^{-1024} \approx 1$ there is a version of the good document that has the same hash as a version of the fraudulent document.

Finding a match, Fred can ask Alice to sign a good version and then append the signature to the fraudulent contract.

BREAKING CRYPTOSYSTEMs and DIGITAL SIGNATURES

- **We say that an encryption system has been broken if one can determine a plaintext from a cryptotext (often).**
- **A digital signature system is considered as broken if one can (often) forge signatures.**
- **In both cases, a more ambitious goal is to find the private key.**

RSA BASED SIGNATURES - CHOICE of PUBLIC EXPONENT

The common choice of a public exponent e is

$$3$$

or

$$2^{16}+1$$

When the value $2^{16}+1$ is used, signature verification requires 17 multiplications, as opposed to roughly 1000 when a random $e \leq O(n)$ is used.

UNDENIABLE SIGNATURES I

Undeniable signatures are signatures that have two properties:

- A signature can be verified only in the cooperation with the signer – by means of a challenge-and-response protocol.
- The signer cannot deny a correct signature. To achieve that, steps are a part of the protocol that force the signer to cooperate – by means of a disavowal protocol – this protocol makes possible to prove the invalidity of a signature and to show that it is a forgery. (If the signer refuses to take part in the disavowal protocol, then the signature is considered to be genuine.)

Undeniable signature protocol of Chaum and van Antwerpen (1989), discussed next, is again based on infeasibility of the computation of the discrete logarithm.

UNDENIABLE SIGNATURES II

Undeniable signatures consist of:

- 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 $\in Z_p^*$ is of order r;
- $1 \leq x \leq r-1$, $y = q^x \mod p$;
- G is a multiplicative subgroup of Z_p^* of order q (G consists of quadratic residues modulo p).

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

Signature: $s = sig_K(w) = w^x \mod p$.

FOOLING and DISALLOWED PROTOCOL I

Since it holds:

Theorem If $s \neq w^x \mod p$, then Alice will accept s as a valid signature for w with probability 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 chooses $e_1, e_2 \in Z_r^*$.
- Alice computes $c = s^{e1} y^{e2} \mod p$ and sends it to Bob.
- Bob computes $d = c^{x^{(-1)} \mod r} \mod p$ and sends it to Alice.
- Alice verifies that $d \neq w^{e1} q^{e2} \pmod p$.
- Alice chooses $f_1, f_2 \in Z_r^*$.
- Alice computes $C = s^{f1} y^{f2} \mod p$ and sends it to Bob.
- Bob computes $D = C^{x^{(-1)} \mod r} \mod p$ and sends it to Alice.

FOOLING and DISALLOWED PROTOCOL II

- Alice verifies that $D \neq w^{f1} q^{f2} \pmod p$.
- Alice concludes that s is a forgery iff

$$(dq^{-e2})^{f1} \equiv (Dq^{-f2})^{e1} \pmod p.$$

CONCLUSIONS

It can be shown:

Bob can convince Alice that an invalid signature is a forgery. In order to do that it is sufficient to show that if $s \neq w^x$, then

$$(dq^{-e2})^{f1} \equiv (Dq^{-f2})^{e1} \pmod p$$

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.