Textbook: Introduction to Cryptography 2nd ed.
By J.A. Buchmann
Chap 12 Digital Signatures

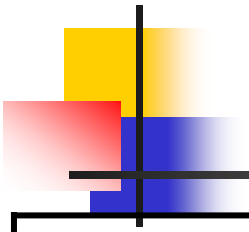
Department of Computer Science and Information Engineering,
Chaoyang University of Technology

朝陽科技大學資工系

Speaker: Fuw-Yi Yang 楊伏夷

伏夷非征番,
道德經 察政章(Chapter 58) **伏**者潛藏也
道紀章(Chapter 14) 道無形象, 視之不可見者曰**夷**

Contents



*I*dea
*S*ecurity
*R*SA Key generation
*O*rder of group elements
*M*ultiplicative group of residues mod *m* – Euler's Theorem
*R*SA assumption
*B*asic--RSA signature generation, verification, example
*A*ttacks on basic scheme
*R*SA signature with redundancy, hash function
*S*ecurity of RSA signature with hash function– random oracle model

Contents

12.1 Idea

Digital signatures are used to sign electronic documents. Such signatures have **properties similar to handwritten signatures**.

If Alice signs a document with her handwritten signature, then everybody who sees the document and who knows Alice's signature can verify that Alice has in fact signed the document.

In many situations, electronic documents also must be signed. For example, *electronic contracts*, *electronic bank transactions*, and *binding electronic mails* must be signed.

In principle, digital signatures work as follows. Suppose that Alice wants to sign the document m. She **uses a secret key d and computes the signature s**. Using the corresponding public key e, Bob can verify that s is in fact the signature of m.

12.1 Idea

Each signature scheme consists of three parts.

The first part is an algorithm for generating the secret and public key.

The second part is an algorithm for generating digital signatures. That algorithm uses the secret key.

The third part is an algorithm for verifying a digital signature. That algorithm uses the public verification key.

12.2 Security

12.2.1 Security of the private key

A digital signature scheme can only be secure if **the problem of computing the secret signature key from publicly available information, in particular, from the public verification key, is intractable.**

The **signature scheme that are used today** have this property. It is based on the intractability of certain computational problems from number theory. However, there is no proof for the intractability of those problems.

12.2 Security

12.2.2 No-message attacks

Finding the secret signature key is not the only possible goal of an attacker. He can also try to **generate new valid signatures without the knowledge of the secret signature key**. This is called an **existential forgery**. To be more precise, the attacker proceeds as follows.

1. The attacker obtains Alice's public verification key.

2. The attacker computes a message x and a signature for x that can be verified with Alice's verification key.

Since no valid signatures of other documents are used, this attack is called a ***no-message attack.*** A signature scheme is called **secure against a no-message attack** if no polynomial time attacker can mount such an attack that is successful with non-negligible probability.

12.2 Security

12.2.3 Chosen message attacks

It is not sufficient that signature scheme is secure against a no-message attacks. It is possible that an attacker knows valid signatures and used them to construct new signatures.

The attack is called the chosen message attack. A signature scheme is called **secure against a chosen message attacks** if no polynomial time chosen message attack is possible that is successful with non-negligible probability.

12.2 Security

12.2.3 Chosen message attacks

The chosen message attack is as follows.

1. The attacker obtains Alice's public verification key.

2. The attacker computes a message x and a signature for x that can be verified with Alice's verification key. During the computation, the attacker can always obtain signatures of documents of his choice.

Example 12.2.2 A web server grants access only to legitimate users. To verify the identity of a user, the Web server asks the user to sign challenge message. If an attacker impersonate the Web server, then he can obtain signatures of documents of his choice.

12.3 RSA Signatures

12.3.1 Key generation

The key generation for RSA signatures:

Alice chooses independently two large random primes p and q and an exponent e with $1 < e < (p - 1)(q - 1)$ and $\gcd(e, (p - 1)(q - 1)) = 1$.

She compute $n = pq$ and $d \in z$ with $1 < d < (p - 1)(q - 1)$ and

$$ed \equiv 1 \bmod \phi(n).$$

Her public key is (n, e) and her secret key d.

See next pages to learn some rationales.

Order of group elements

Let G be a group that is multiplicatively written with neutral element 1.

Definition 2.9.1 Let $g \in G$. If there is a positive integer e with $g^e = 1$, then the smallest such integer is called the *order* of g in G. Otherwise, we say that the order of g in G is infinite.

multiplicative group: construct and explain in the follows.

Definition 2.1.1 m is a positive integer, a and b are integers.
We say that a is congruent to b modulo m, and we write $a \equiv b \bmod m$, if m **divides** the $b - a$.

Example 2.1.2

We have $-2 \equiv 19 \bmod 21$, $10 \equiv 0 \bmod 2$.

Order of group elements

The equivalent class of a consists of all integers that are obtained from a by adding integer multiples of m; i.e., $\{b: b \equiv a \bmod m\} = a + mZ$.
The equivalent class is called **residue class of a mod m**.

The **set of residue classes mod m** is denoted by Z/mZ.

Example 2.8.2 The multiplicative group of residues modulo 12 is $(Z/12Z)^* = \{ 1 + 12Z, 5 + 12Z, 7 + 12Z, 11 + 12Z\}$.
Its *order* is $\varphi(12) = 4$.

Multiplicative group of residues mod m

The group of **invertible** residue classes modulo m is called the *multiplicative group of residues modulo m* and is written $(Z/mZ)^*$.

Its *order* is denoted by $\phi(m)$.

The function

$$N \to N,\ m \to \phi(m)$$

is called the Euler ϕ *-function.*

$\phi(m)$ is the number of integers a in $\{1,2,\ldots,m\}$ with $\gcd(a, m) = 1$.

Euler's Theorem
(K.H.Rosen, Elementary Number Theory 4th Ed., p217)

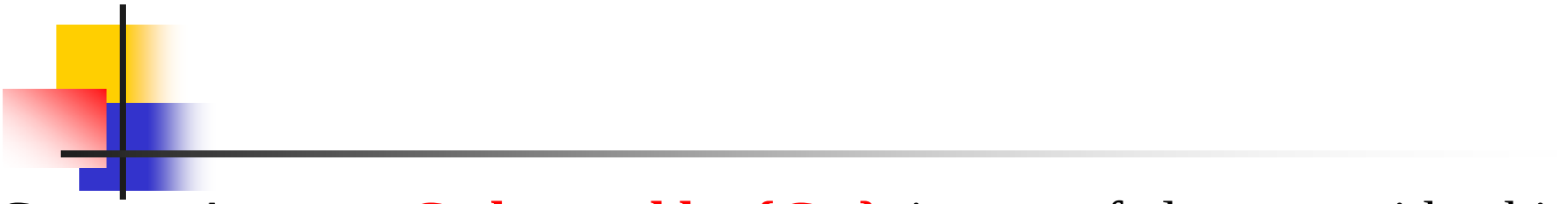
Definition

Let n be a positive integer. The *Euler phi-function* $\phi(n)$ is defined to be the number of positive integers not exceeding n that are relatively prime to n.

Euler's Theorem.

If m is a positive integer and a is an integer with $\gcd(a, m) = 1$, then $a^{\phi(m)} = 1 \bmod m$.

The Finite Field – Groups 1/2

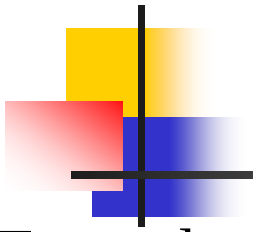


Groups: A **group *G*, denoted by {*G*, ·},** is a set of elements with a binary operation · such that:

(A1) Closure: $a, b \in G$ implies that $a \cdot b \in G$
(A2) Associative: $a, b, c \in G$ implies that $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
(A3) Identity: For all a in G,
there is an element e in G s.t. $a \cdot e = e \cdot a$
(A4) Inverse: For all a in G, there exists
an element b (i.e. a^{-1}) in G, s.t. $a \cdot b = e$

Abelian group: (A5) Commutative law: $a \cdot b = b \cdot a$ for all a, b in G

The Finite Field – Groups -2/2



Example: $\{Z_7^*, \cdot\}$ or written as $(Z/7Z)^*$

$G = \{1, 2, 3, 4, 5, 6\}$, $\cdot$ is modular multiplication (mod 7)

$5 \cdot 3 \bmod 7 = 1$, $(5^{-1} = 3, 3^{-1} = 5)$

$2 \cdot 6 \bmod 7 = 5$,

$6 \cdot 2 \bmod 7 = 5$, (Abelian group)

Example 2.3.5

The additive group Z has infinite order. Also $\{Z, +\}$

The additive group Z/mZ has order m. Also $\{Z_m, +\}$

Order of group elements

Theorem 2.9.2 Let $g \in G$ and $e \in Z$. Then $g^e = 1$ **iff** e is divisible by the order of g in G.

Proof. Let n be the *order* of g in G.

If $e = kn$, then $g^e = g^{kn} = (g^n)^k = 1^k = 1$.

Conversely, let $g^e = 1$ and $e = qn + r$ with $0 \le r < n$.

Then $g^r = g^{e-qn} = g^e g^{-nq} = 1$. ⇨ $r = 0$. ⇨ $n \mid e$.

Order of group elements

Example for Theorem 2.9.1, Theorem 2.9.2

$2^1 \bmod 7=2$	$3^1 \bmod 7=3$	$4^1 \bmod 7=4$	$5^1 \bmod 7=5$	$6^1 \bmod 7=6$
$2^2 \bmod 7=4$	$3^2 \bmod 7=2$	$4^2 \bmod 7=2$	$5^2 \bmod 7=4$	$6^2 \bmod 7=1$
$2^3 \bmod 7=1$	$3^3 \bmod 7=6$	$4^3 \bmod 7=1$	$5^3 \bmod 7=6$	
	$3^4 \bmod 7=4$		$5^4 \bmod 7=2$	
	$3^5 \bmod 7=5$		$5^5 \bmod 7=3$	
	$3^6 \bmod 7=1$		$5^6 \bmod 7=1$	

Security of RSA Cryptosystem--RSA assumption

Assume it is given a RSA modulus n, an exponent e, and an element z selected randomly from the group Z_n^* ($z \in Z_n^*$).

The **RSA problem** is to find the eth root of z, *i.e.* finding $y \in Z_n^*$ such that $z = y^e \bmod n$ or $y = z^{1/e} \bmod n$.

The **RSA assumption** says that it is hard to solve the RSA problem.

12.3 RSA Signatures

12.3.2 Signature generation

Assume that Alice wants to sign $m \in \{0, 1, \ldots, n\text{-}1\}$.

The integer m can be a short document or message.

Alice computes $\boldsymbol{s \equiv m^d \bmod n}$.

The **signature** is s.

12.3 RSA Signatures

12.3.3 Verification

Assume that Bob wants to verify the signature s.

He gets Alice's public key (n, e) from some public directory and recovers the signed message by computing

$$\boldsymbol{m \equiv s^e \bmod n}. \text{ sign } m \in \{0, 1, \ldots, n\text{-}1\}.$$

He does not need to know m in advance. But he is sure that Alice has generated s.

12.3 RSA Signatures
12.3.3 Verification

Example 12.3.1 Alice chooses $p = 11$, $q = 23$, $e = 3$.

She obtains $n = 253$, $d = 147$.

Alice's public key is (253, 3). Her secret key is 147.

To sign on message 111. She computes $s = 111^{147} \bmod 253 = 89$.

Anyone verifies the signature by computing

$$m = 89^3 \bmod 253 = 111.$$

The story may be that Alice wants to obtain $ 111 from an automated teller machine. She signs 111 and sends the resultant signature to the ATM. The cash dispenser verifies the signature.

12.3 RSA Signatures

12.3.3 Verification—find inverse

Use the Extended Euclid's algorithm to find the inverse of an integer:

1. Use the Euclid's algorithm to find the greatest common divisor of x and p.
2. Then work backs to write 1 as an integer linear combination of x and p.

$\phi(n) = (p-1)(q-1) = 220$

$220 = 3 \cdot 73 + 1 \Rightarrow 1 = 220 - 3 \cdot 73 \Rightarrow 220 - 3 \cdot 73 \equiv 1 \bmod 220$

$\Rightarrow -3 \cdot 73 \equiv 1 \bmod 220 \Rightarrow (-73) \cdot 3 \equiv 1 \bmod 220$

$\Rightarrow 147 \cdot 3 \equiv 1 \bmod 220$

12.3.3 Verification—find inverse

Introduction to Algorithms T.H. Cormen et al.

Euclid's algorithm: Euclid(a, b)

```
{ if b = 0 then return a
        else return Euclid(b, a mod b)
}
```

Extended-Euclid's algorithm: Extended-Euclid(a, b)

```
{ if b = 0 then return (a, 1, 0)
(d', x', y') ← Extended-Euclid(b, a mod b)
(d, x, y) ← (d', y', x' - ⌊a/b⌋)
return (d, x, y)
 }
```

12.3 RSA Signatures
12.3.4 Attacks

1. If the attacker, Oscar, is able to replace Alice's public key with his own public key without verifier's noticing, then he can sign in Alice's name.

2. no-message attack: Oscar chooses an integer $s \in \{0, 1, \ldots, n\text{-}1\}$. Then he claims that s is an RSA signature of Alice. If $\boldsymbol{m \equiv s^e \bmod n}$ is a meaningful text, then Oscar was able to fake a signature of Alice. **See Example 12.3.2 in next page.**

3. Another danger comes from the fact that RSA is multiplicative.

If $m_1, m_2 \in \{0, 1, \ldots, n\text{-}1\}$ and $s_1 \equiv m_1^d \bmod n$ and $s_2 \equiv m_2^d \bmod n$ are signatures of m_1 and m_2, then $s = s_1 s_2 = (m_1 m_2)^d \bmod n$ is the signature of $m = m_1 m_2 \bmod n$. **other attacks in next pages.**

12.3 RSA Signatures
12.3.4 Attacks

Example 12.3.2 Alice's public key is (253, 3). Her secret key is 147.

Oscar wants to withdraw money from Alice's account.

He sends $s = 123$ to the cash dispenser. The cash dispenser computs

$$m = 123^3 \bmod 253 = 52.$$

It believe that Alice wants to withdraw $ 52, but this is not true.

Alice has never signed the $ 52. She was the victim of an existential forgery.

12.3 RSA Signatures
12.3.4 Attacks

Example of Chosen message attack on RSA As shown previously, from two valid RSA signatures, a third one can be computed. An attacker can use the multiplicativity of RSA signatures to forge a valid signature for any document.

Let $m \in \{0, 1, \ldots, n\text{-}1\}$ be a message.

Attacker selects a message $m_1 \in \{0, 1, \ldots, n\text{-}1\}$ with $\gcd(m, m_1) = 1$.

Compute $m_2 = m\, m_1^{-1} \bmod n$. Two valid signatures $s_1 \equiv m_1^d \bmod n$ and $s_2 \equiv m_2^d \bmod n$ are obtained.

Then the attacker computes $s = s_1 s_2 = (m_1 m_2)^d = (m)^d \bmod n$.

12.3 RSA Signatures

12.3.5 Signature with redundancy

Two of the attacks of 12.3.4 are impossible if only integers $m \in \{0, 1, \ldots, n\text{-}1\}$ having a binary expansion of the form ww with $w \in \{0, 1\}^*$ can be signed. Thus the binary expansion has two identical halves.

The function

$$R: \{0, 1\}^* \rightarrow \{0, 1\}^*, w \rightarrow R(w) = ww,$$

which is used for the generation of the special structure of the documents that can be signed, is called a ***redundancy function***.

Clearly, other redundancy functions can also be used.

12.3 RSA Signatures

12.3.6 Signature with hash functions

If Alice wants to sign an arbitrarily long document x, then she uses a publicly known collision resistant hash function

$$h: \{0, 1\}^* \rightarrow \{0, 1, \ldots, n\text{-}1\}.$$

Since h is collision resistant, h is also a one-way function.

In practice, h is constructed using a standard collision resistant hash function whose values are, for example, 160 bitstrings.

The signature of the document x is

$$s = h(x)^d \bmod n.$$

Alice sends **s together with the document x** to Bob.

Boy verifies the signature by checking **whether $h(x) = s^e \bmod n$**.

The multiplicatively attacks described previously can longer work.

Notation in M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

The Paradigm: Suppose there is a protocol problem Π.

In order to devise a good protocol P for Π:

1. Find a formal definition for Π in the model of computation in which all parties (including the adversary) share a random oracle R.
2. Devise an efficient protocol P for Π in this random oracle model.
3. Prove that P satisfies the definition for Π.
4. Replace oracle accesses to R by computation of h.

M. Bellare and P. Rogaway, Random oracles are practical:
a paradigm for designing efficient protocols

Notation is as follows.

$A(x, y, \ldots)$ refers to the probability space which to the string σ assigns the probability that A, on input x, y, …, outputs σ, where A is a **probabilistic algorithm** with inputs x, y, ….

M. Bellare and P. Rogaway, Random oracles are practical:
a paradigm for designing efficient protocols

1. A probability Space is a triple (S, $\mathcal{S}$, P) on the domain S, where (S, $\mathcal{S}$) is a measurable space, $\mathcal{S}$ are the measurable subsets of S, and P is a measure on $\mathcal{S}$ with $P(\mathcal{S}) = 1$.

2. A probability space (Ω, F, P) is a measure space with a measure P that satisfies the probability axioms.

Ω is a nonempty set whose elements are known as **outcomes** or **states of nature.**

F is a subset of Ω, its elements are called events, which are sets of outcomes for which one can ask a probability.

P is a **probability measure** which is a function from F to the real numbers that assigns to each event a *probability* between 0 and 1. It must satisfy the probability axioms.

M. Bellare and P. Rogaway, Random oracles are practical:
a paradigm for designing efficient protocols

If the space concerns one flip of a fair coin, then the outcomes are heads and tails: $\Omega = \{H, T\}$

The events are {H}: heads, {T}: tails, {}: neither heads nor tails, and {H,T}: heads or tails.

So, $F = \{\{H\},\{T\},\{\},\{H,T\}\}$

There is a fifty percent chance of tossing either heads or tail: $P(\{H\}) = P(\{T\}) = 0.5$. The chance of tossing neither is zero: $P(\{\})=0$, and the chance of tossing one or the other is one: $P(\{H,T\})=1$.

M. Bellare and P. Rogaway, Random oracles are practical:
a paradigm for designing efficient protocols

If S is a **probability space**, then its **support** (the set of elements of positive probability) is denoted by $[S]$.

If S is a probability space then $x \leftarrow S$ denotes the **algorithm** which assigns to x an element randomly selected according to S.

For probability space S, T,…, the notation
$Pr[x \leftarrow S; y \leftarrow T; \ldots : p(x, y, \ldots)]$ denotes the **probability** that the **predicate** $p(x, y, \ldots)$ is true after the execution of the algorithms $x \leftarrow S$, $y \leftarrow T$, *etc.*

$\{x \leftarrow S; y \leftarrow T; \ldots : f(x, y, \ldots)\}$ denotes the **probability space** which to the string σ assigns the probability
$Pr[x \leftarrow S; y \leftarrow T; \ldots : \sigma = f(x, y, \ldots)]$.

M. Bellare and P. Rogaway, Random oracles are practical:
a paradigm for designing efficient protocols

<***a, b,…:*** *x* ← *S*; *y* ← *T*; …: ***f(a, b,…,x, y,…)***> denotes the algorithm which on input *a*, *b*,… runs the sequence of **experiments** *x* ← *S*, *y* ← *T*,…, and outputs *f*(*a*, *b*,…,*x*, *y*,…).

A **random oracle** *R* is a map from $\{0, 1\}^*$ to $\{0, 1\}^\infty$ chosen by selecting each bit of *R*(*x*) uniformly and independently, for every *x*. *H*: $\{0, 1\}^* \rightarrow \{0, 1\}^k$ denotes a **hash function**.

Trapdoor permutation generator is a *PPT* algorithm *G* which on input 1^k outputs a triple of algorithms (f, f^{-1}, d). The first two are deterministic and the last is probabilistic. We require that [$d(1^k)$] be a subset of $\{0, 1\}^k$ and that f, f^{-1} be permutation on [$d(1^k)$] which are inverse of one another.

12.3.6 Signature with hash functions – Security 1/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

A digital signature scheme is a triple (*GS*, *Sign*, *Verify*) of polynomial time algorithms, called the generator, signing algorithm, and verifying algorithm, respectively.

The first two are probabilistic and the last two have access to the random oracle.

A **random oracle** R is a map from $\{0, 1\}^*$ to $\{0, 1\}^\infty$ chosen by selecting each bit of $R(x)$ uniformly and independently, for every x.

H: $\{0, 1\}^*$ to $\{0, 1\}^k$ denotes a hash function.

On input 1^k, the generator produces a pair (PK, SK) of matching public and secret keys. E.g. ((e, n), d)

12.3.6 Signature with hash functions – Security 2/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

To sign message m compute $\sigma \leftarrow Sign^R(SK, m)$;
to verify (m, σ) compute $Verify^R(PK, m, \sigma) \in \{0, 1\}$.

It must be the case that $Verify^R(PK, m, \sigma) = 1$
for all $\sigma \in [Sign^R(SK, m)]$.

An *S*-adversary is a non-uniform polynomial-time algorithm ***F*** which access to *R* and a signing oracle.

12.3.6 Signature with hash functions – Security 3/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

The output of F is a pair (m, σ) such that m was not queried of the **signing oracle**. The signature scheme is secure if for every S-adversary F the function $\varepsilon(k)$ defined by

$Pr[R \leftarrow 2^{\infty}; (PK, SK) \leftarrow GS(1^k);$

$(m, \sigma) \leftarrow (PK)$: $Verify^R(PK, m, \sigma) = 1] \leq k^{-\omega(1)}$.

$k^{-\omega(1)}$: any non-negligible function

F is successful if its output (m, σ) satisfies $Verify^R(PK, m, \sigma) = 1$.

12.3.6 Signature with hash functions – Security 4/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

Scheme. A **signature** is obtained by computing

$\sigma = f^{-1}(H(m)) \leftarrow Sign^H(f^{-1}, m)$, and **verified** by

$H(m) = f(\sigma)$, where H: $\{0, 1\}^* \rightarrow \{0, 1\}^k$.

The scheme is secure against **adaptive chosen message attack**.

Proof. Assume that F is an S-adversary successful with a probability $\lambda(k)$ that is not negligible. We construct algorithm $M(f, d, y)$ so that

$\varepsilon(k) = Pr[(f, f^{-1}, d) \leftarrow G(1^k); x \leftarrow d(1^k); y \leftarrow f(x): M(f, d, y) = x]$.

Let $PK = f$. It flips coins for F and starts running $F(PK)$.

12.3.6 Signature with hash functions – Security 5/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

Assume that F makes exactly $n(k)$ queries to H, all distinct, and that if F makes a signing query m than it has already queried $H(m)$.

M chooses $t \in \{1, 2, \ldots, n(k)\}$ at random.

M simulates F as shown in the table below.

12.3.6 Signature with hash functions – Security 6/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

Assume that we are given a public key (n, e). We want to find **the inverse of a random number $y \in Z_n^*$**.

Argument m_i	M's answer
F makes i_{th} H query with parameter m_i.	If $i = t$, M answers y. *I.e.*, $H(m_t) = y$.
	Otherwise, answers $f(r_i)$, where $r_i \leftarrow \{0, 1\}^k$. *I.e.*, $H(m_i) = f(r_i)$.
F makes signing query with parameter $m \in \{m_1, \ldots, m_{n(k)}\}$.	If $m = m_t$, M halts, admitting failure. (M cannot compute $f^{-1}(y)$).
	Otherwise, answers r_i where $i \neq t$ satisfies $m = m_i$.

12.3.6 Signature with hash functions – Security 7/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

Let (m, σ) be the output of F. If $m \neq m_t$, then M cannot uses (m, σ) to obtain $f^{-1}(y)$.

Otherwise, $H(m) = y = f(\sigma)$. Namely, $f^{-1}(y) = \sigma$, F has computed $f^{-1}(f(r_i))$.

Let *SK* be the event that *F* is successful in this experiment. Then

***M* succeeds in computing $f^{-1}(y)$ with probability *Pr*[*SK* and $m = m_t$].**

If $m \notin \{m_1, \ldots, m_{n(k)}\}$ then *F* succeeds with probability at most 2^{-k}

(*F* merely guesses the value of σ).

12.3.6 Signature with hash functions – Security 8/8

M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols

Thus F succeeds in these experiments with the following probability:

$$\lambda(k) - 2^{-k} = \sum_{i=1}^{n(k)} \Pr[SK \text{ and } m = m_i] .$$

Since M chooses t in random, thus the following equation holds true.

$$(\lambda(k) - 2^{-k})/n(k) \le Pr[SK \text{ and } m = m_t] = \varepsilon(k).$$

Namely, ***M* succeeds in computing the inverse of *f* at *y*** with non-negligible probability if F succeeds in forging a signature with non-negligible probability $\lambda(k)$.

This result contradicts the assumption of difficulty of computing f^{-1}.

Thus F cannot succeed in forging a signature with non-negligible probability.

12.4 Signatures from Public-Key Systems

Consider another public-key cryptosystem. For a pair (e, d) of public key and corresponding private key, let **E_e be the encryption function** and let **D_d be the decryption function**.

Suppose that for any such pair (e, d) and any plaintext m, we have

$$m = E_e(D_d(m)).$$

Then a signature scheme can be constructed from this public-key system. The signature of the document m is

$s = D_d(h(m)m))$, where h is a public known collision resistant hash function. This signature is verified by computing $h(m) = E_e(s)$. It is also possible to use a redundancy function instead of the hash function. The details are explained in the standard ISO/IEC 9796.

12.5 ElGamal Signature

The **ElGamal signature scheme** is similar to the ElGamal cryptosystem, although it is not constructed from it by the method described in Section 12.4.

Its security is based on the difficulty of computing discrete logarithms in $(Z/pZ)^*$, where p is a prime number.

12.5 ElGamal Signature

12.5.1 Key generation

Alice chooses a prime number p and a primitive root g mod p whose order mod p is sufficiently high.

Then he chooses a random exponent $a \in \{1, \ldots, p\text{-}2\}$ and computes $A = g^a \bmod p$.

The **public key** of Alice is (p, g, A).
The **secret key** is the exponent a.

12.5 ElGamal Signature

12.5.2 Signature generation

Alice signs a document $x \in \{0, 1\}^*$. She uses the publicly known collision resistant hash function

$h: \{0, 1\}^* \rightarrow \{1, \ldots, p\text{-}2\}$.

Alice chooses a random number **$k \in \{1, \ldots, p\text{-}2\}$ which is prime to p-1**. She computes

$r = g^k \bmod p$, and

$s = k^{-1}(h(x) - ar) \bmod p\text{-}1$.

The signature of x is (r, s). The signed message is $((r, s), x)$.

12.5 ElGamal Signature

12.5.3 Verification ($r = g^k \bmod p$, $s = k^{-1}(h(x) - ar) \bmod p\text{-}1$)

The verifier, Bob, uses Alice's public key (p, g, A). As in the RSA signature scheme, he has to convince himself of the authenticity of this public key. **He verifies that**

$1 \le r \le p\text{-}1$ and

$A^r r^s = g^{h(x)} \bmod p$.

Example 12.5.1 Alice chooses $p = 23$, $g = 7$, $a = 6$ and computes $A = g^a \bmod p = 4$. Her public key is ($p = 23$, $g = 7$, $A = 4$), private key is $a = 6$. Alice wants to sign the document x, which has value $h(x) = 7$.

She chooses $k = 5$ and obtains $r = 17$, $k^{-1} \bmod p\text{-}1 = 9$.

Therefore, $s = k^{-1}(h(x) - ar) = 9\ (7 - 6 * 17) \bmod 22 = 3$.

The signature is (17, 3).

12.5 ElGamal Signature

12.5.4 The choice of *p*

If the attacker, Oscar, can compute discrete logarithms mod *p*, then he can determine Alice's secret key and can generate signatures in Alice's name.

This remains the only known general method of generating ElGamal signatures. Therefore, ***p* must be chosen such that computing discrete logarithms mod *p* is infeasible**.

Given the discrete logarithms known today, this means that ***p* should be at least a 768-bit number**.

Also, **primes of special forms** for which certain DL algorithms such as the **Pohlig-Hellman method** (Section 10.5) are particular efficient must be avoided. The best strategy is to use random primes.

12.5 ElGamal Signature

12.5.4 The choice of *p*

It is also **dangerous if *p* = 3 mod 4, the primitive root *g* divides *p*-1,** and computing discrete logarithms in the subgroup of $(Z/pZ)^*$ of order g is possible. Therefore, g should not divide p-1.

See next page

12.5 ElGamal Signature

12.5.4 The choice of p

Exercise 12.9.5 Let p be a prime number, $p = 3 \bmod 4$.

Let g be a primitive root mod p and let $A = g^a \bmod p$ be Alice's public key.

Let **g be a divisor of $(p - 1)$ (i.e. $p - 1 = gq$ with $q \in Z$) and $z \in Z$ with $g^{qz} = A^q \bmod p$. ($z = \log_{gq}(A^q)$ can be computed)**

Prove that for each document m the pair $(q, s = (p - 3)(h(m) - qz)/2)$ is a valid ElGamal signature of m. How can this attack prevented?

12.5 ElGamal Signature

12.5.4 The choice of p

Proof. By the verification congruence, $A^r\, r^s = g^{h(x)} \bmod p$, we want to prove that $A^q\, q^s = g^{h(m)} \bmod p$.

From the new signature (q, s) on message m,

we have $A^r\, r^s = A^q\, q^{(p-3)(h(m)-qz)/2} \bmod p$.

Also, $p - 1 = gq \Rightarrow$ **$q = -(g^{-1}) \bmod p$**, **$q^{-1} = -g \bmod p$**,

g is a primitive root mod $p \Rightarrow$ **$(g^{(p-1)/2}) = -1 \bmod p$**.

Therefore,

$$(q^{(p-3)/2}) = (q^{-1}q^{(p-1)/2}) = (-g)\,[(-1)^{(p-1)/2}(g^{-1})^{(p-1)/2}]$$
$$= (-g)\,(-1)\,(g^{(p-1)/2})^{-1} = g \bmod p. \quad (p = 3 \bmod 4)$$
$$A^r\, r^s = A^q\, q^{(p-3)(h(m)-qz)/2} = A^q\, g^{(h(m)-qz)}$$
$$= A^q\, g^{h(m)}\, A^{-q} \quad (z \in (a + Z/(p-1)Z)$$
$$= g^{h(m)} \bmod p.$$

end of proof

12.5 ElGamal Signature

12.5.5 The choice of *k*

For the security of ElGamal signature, **every new signature a new exponent *k* must be chosen**. That is guaranteed if *k* is a random number.

Example 12.5.new If the random number *k* has been used twice in generating signatures $s_1 = k^{-1}(h(x_1) - ar) \bmod p\text{-}1$, and

$s_2 = k^{-1}(h(x_2) - ar) \bmod p\text{-}1$, then the signer's secret signature key, *a*, is solvable.

12.5 ElGamal Signature

12.5.6 Existential forgery no-message attack

Goals of the adversary may be:

Total break -- Disclosing the private key of the signer. It is the most serious attack;

Universal forgery -- Constructing an efficient algorithm which is able to sign messages with good probability of success.

Existential forgery -- providing a new message-signature pair. The corresponding security level is called existential unforgeability.

Means available to the adversary may be:

No-message attack -- the attacker only knows the signer's public key.

Known-message attack -- the attacker has access a list of valid message-signature pairs.

Adaptive-message attack -- the attacker can ask the signer to sign any message of his choice.

12.5 ElGamal Signature

12.5.6 Existential forgery no-message attack

If no hash function is used in the ElGamal signature system, then **existential forgery is possible.**

Without a hash function, the verification congruence is

$$A^r r^s = g^x \bmod p.$$

We show how r, s, x can be chosen such that this congruence is satisfied. **To mount the existential forgery**, **Oscar chooses two integers u, v with $\gcd(v, p\text{-}1) = 1$.** Then he sets

$$r = g^u A^v \bmod p,\ s = -rv^{-1} \bmod p\text{-}1,\ x = su \bmod p\text{-}1.$$

With those values for r and s, the verification congruence

$$A^r r^s = A^r g^{us} A^{vs} = A^r g^{us} A^{-r} = g^x \bmod p.$$

12.5 ElGamal Signature

12.5.6 Existential forgery

Forgery from old signatures

This procedure also works if a collision resistant hash function is used. **But since the hash function is a one-way function**, it is impossible for Oscar to find a document x such that the signature generated is the signature of x.

The condition $1 \leq r \leq p\text{-}1$ is also crucial. If it is not required, then it is possible to generate new signatures from old signatures. We explain.

See next page

12.5 ElGamal Signature

12.5.6 Existential forgery

Forgery from old signatures

Let (r, s) be the ElGamal signature of the document x. **Oscar wants to forge Alice's signature on document x'.**

Given (r, s) and $A^r r^s = g^{h(x)} \bmod p$,

the goal is to compute (r', s') to satisfy $A^{r'} r^{s'} = g^{h(x')} \bmod p$.

1. $u = h(x')h(x)^{-1} \bmod (p - 1)$,
2. $s' = su \bmod (p - 1)$,
3. Using CRT, compute r': $r' = ru \bmod (p - 1)$, $r' = r \bmod p$,
4. The signature on document x' is (r', s').
5. **If $1 \le r' \le p\text{-}1$ and $r' = r \bmod p$, then $r' = ru \ne r \bmod (p - 1)$, since $u = h(x')h(x)^{-1} \ne 1 \bmod (p - 1)$.**

12.5 ElGamal Signature
12.5.7 Efficiency

Signature generation: $r = g^k \bmod p$, and

$$s = k^{-1}(h(x) - ar) \bmod p\text{-}1.$$

Signature verification: $A^r r^s = g^{h(x)} \bmod p$.

By simultaneous exponentiations: $A^r r^s g^{-h(x)} = 1 \bmod p$.

12.5 ElGamal Signature

12.5.8 Secure ElGamal signatures

12.5.9 Generalization

12.5.8 **D. Pointcheval and J. Stern**, Security arguments for digital signatures and blind signatures, Journal of Cryptology, Vol. 13, pp. 361-396, 2000

12.5.9

Like the ElGamal cryptosystem, the ElGamal signature scheme can also **be implemented in any cyclic group whose order is known.**

The implementation, including the security considerations, can be deduced from the implementation in $(Z/pZ)^*$.

2.6 The Digital Signature Algorithm (DSA)

The **Digital Signature Algorithm** (DSA) has been suggested and standardized by the National Institute of Standards and Technology (NISI) of the U.S.

It is an **efficient variant of the ElGamal signature** scheme.

The number of modular exponentiations in the verification is **reduced from three to two** and, more importantly, the number of digits in the exponents is **160** while in the ElGamal signature scheme the exponents have as many bits as the prime p (i.e., at least 768 bits)

12.6 The Digital Signature Algorithm (DSA)

12.6.1 Key generation

Alice chooses a **prime number *q*** with $2^{159} < q < 2^{160}$.

Hence, q has binary length 160.

Alice also chooses a **large prime *p*** with the following properties:

1. $2^{511 + 64t} < p < 2^{512 + 64t}$ for some $t \in \{0, 1, \ldots, 8\}$,
2. the prime number q, which was chosen first, divides p-1.

The binary length of p is between 512 and 1024 and is a multiple of 64. Therefore, the binary expansion of p is a sequence of 8 to 16 bit-strings of length 64. **The condition $q \mid (p - 1)$ implies that the group $(Z/pZ)^*$ contains elements of order q** (Theorem 2.21.1).

12.6 The Digital Signature Algorithm (DSA)

12.6.1 Key generation

Next, Alice chooses a **primitive root** x mod p and computes

$$g = x^{(p-1)/q} \bmod p.$$

Then $g + pZ$ has order q in $(Z/pZ)^*$.

Finally Alice chooses a random exponent $a \in \{1, \ldots, q-1\}$ and computes $A = g^a \bmod p$.

The **public key** of Alice is (p, q, g, A).

The **secret key** is the exponent a.

Note that the residue class $A + pZ$ is an element of the subgroup generated by $g + pZ$.

12.6 The Digital Signature Algorithm (DSA)

12.6.2 Signature generation

Alice signs a document $x \in \{0, 1\}^*$. She uses the publicly known collision resistant hash function

$$h: \{0, 1\}^* \rightarrow \{1, 2, \ldots, q\text{-}1\}.$$

Alice chooses a random number $\boldsymbol{k \in \{1, 2, \ldots, q\text{-}1\}}$. She computes

$r = (g^k \bmod p) \bmod q$, and

$s = k^{-1}(h(x) + ar) \bmod q$.

The signature of x is (r, s). The signed message is $((r, s), x)$.

12.6 The Digital Signature Algorithm (DSA)

12.6.3 Signature verification

The verifier, Bob, uses Alice's public key (p, q, g, A) and the public hash function . **He verifies that**

$1 \le r \le q\text{-}1$, $1 \le s \le q\text{-}1$ and

check whether $r = (g^{(s^{-1}h(x)) \bmod q} A^{(rs^{-1}) \bmod q} \bmod p) \bmod q$.

$r = (g^k \bmod p) \bmod q$, and

$s = k^{-1}(h(x) + ar) \bmod q. \Rightarrow k = s^{-1}(h(x) + ar) \bmod q.$

12.6 The Digital Signature Algorithm (DSA)

12.6.4 Efficiency

1. Signature generation

$r = (g^k \bmod p) \bmod q$, and

$s = k^{-1}(h(x) + ar) \bmod q$.

pre-computation of modular exponentiation makes the signature generation much faster.

2. Signature verification

check whether $r = (g^{(s^{-1}h(x)) \bmod q} A^{(rs^{-1}) \bmod q} \bmod p) \bmod q$.

only two exponentiations mod p are required.

The bit length of exponent is only 160.

12.6 The Digital Signature Algorithm (DSA)

12.6.5 Security

As in the ElGamal signature scheme, it is necessary to choose a **new random exponent *k* for each new signature** (see Section 12.5.5).

Moreover, the use of hash function and **checking conditions $1 \le r \le q-1$, $1 \le s \le q-1$ is mandatory to prevent possible existential forgery** (see Section 12.5.6).

The only known general attack against DSA is **computation of secret key *a* from the given public key *A***.

For efficiency reasons, g is chosen to generate a subgroup H of $(Z/pZ)^*$, i.e., $A \in H = \langle g \rangle \subset (Z/pZ)^*$. **Thus the security of DSA depends on the difficulty of computation of discrete logarithm in the subgroup *H*.**

12.6 The Digital Signature Algorithm (DSA)

12.6.5 Security

In principle, there are two methods of computing discrete logarithms in H.

The first is to apply an **index calculus algorithm** in Z/pZ (see Section 10.6). But it is unknown how index calculus algorithms can take advantage of the fact that a discrete logarithm in a subgroup of $(Z/pZ)^*$ is to be computed.

The **running time of all known index calculus algorithms depends on the size of the prime number *p***, but p is chosen such that index calculus attacks are infeasible.

12.6 The Digital Signature Algorithm (DSA)

12.6.5 Security

The **second** possibility is to apply a **generic method that works for all cyclic groups**. The most efficient generic methods in groups of prime order are due to **Shanks and Pollard** (Section 10.3 and 10.4). In a group of order q, they required more than $(q)^{1/2}$ group operations. Since $q > 2^{159}$, this is infeasible with current technology.

12.7 Undeniable Signature

Menezes , Oorschot, and Vanstone give the following example for an application of an **undeniable signature**.

A customer wishes to gain access to a safe-deposit box room in a bank. The bank requires the customer to sign a time and date document before access is granted.

However, the customer does not want the bank to be able to tell anyone when he has actually used those facilities. Therefore, he uses an undeniable signature in which verification is impossible without his direct involvement.

12.7 Undeniable Signature

In such an undeniable signature protocol it is possible that **a document is signed and that the signer later tries to deny** that she has signed that document.

However, **this is made impossible by the protocol** and this explains the notion “undeniable signature”.

12.7 Undeniable Signature

12.7.1 Specification

The **participants** of an undeniable signature protocol are **the signer and the verifier**. The signer has a secret signing key. The verifier has access to the signer's public verification key. The protocol has the following steps.

1. Key generation
2. Signature generation
3. Signature verification protocol and disavowal protocol

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

We present a generalized version of the Chaum-Van Antwerpen scheme.

1. Key generation. The signer chooses a finite cyclic group G of known prime order q, a generator g of that group, and a random nonzero exponent $a \in Z_q$. She compute $A = g^a$.

The **public verification key** is (G, g, q, A). The **private key** is a.

The signer also selects a cryptographic hash function $h:\{0, 1\}^* \rightarrow G$ and publishes that hash function.

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

2. Signature generation.

Given a document $m \in G$ and her signing key a, the signer computes the signature $s = h(m)^a$.

So a valid signature of a document is a $(\log_g A)$th power of the hash value of that document.

Clearly, without additional help, nobody can verify such a signature since the discrete logarithm of A to the base g is secret.

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

3. Signature verification protocol and disavowal protocol.

In the **verification process**, the signer convinces the verifier that ***s* is the ($\log_g A$)th power $h(m)$.**

3.1 The verifier obtains the document, the signature *s*, and the signer's verification key.

3.2 Challenge: The verifier selects random integers $u, v \in Z_q$. He computes $z = s^u A^v = h(m)^{ua} g^{va}$. The verifier sends the challenge to the signer.

3.3 Response: The signer computes the *a*th root *w* of *z*. She does this by computing the inverse a^{-1} of *a* mod *q* and by determining $w = z^{a^{-1}}$. She sends the response *w* to the verifier.

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

3.4 Verification: The verifier compares w to $h(m)^u g^v$ which is the ath root of z. If this two group elements are equal, then the verifier accepts the signature. Otherwise he rejects it.

	Signer		Verifier
Signature generation	$s = h(m)^a$		
Signature verification	$w = z^{a^{-1}}$	$\leftarrow z$ $w \rightarrow$	$u, v \in_R Z_q$, $z = s^u A^v$ $w =? h(m)^u g^v$ Accept if it is equal.

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

Lemma 12.7.1.

1. For any $z \in G$, there are q pairs (u, v) such that $z = s^u A^v$.

2. If $s \neq h(m)^a$ and if $z \in G$, then for each $w \in G$ there is exactly one pair (u, v) such that $w = h(m)^u g^v$ and $z = s^u A^v$.

Proof.

1. Let $j = \log_g z$, $k = \log_g h(m)$, $l = \log_g s$.

Then the equation $z = s^u A^v$ can be written as $g^j = g^{ul + va}$.

Since g generates G (G is with prime order q) this yields the linear congruence $lu + av = j \bmod q$.

This congruence has exactly q solutions $(u, v) \in Z_q \times Z_q$.

*endofproof*1

12.7 Undeniable Signature
12.7.2 Chaum-van Antwerpen scheme

Lemma 12.7.1.

2. If $s \neq h(m)^a$ and if $z \in G$, then for each $w \in G$ there is exactly one pair (u, v) such that $w = h(m)^u g^v$ and $z = s^u A^v$.

Proof. 2. Let $j = \log_g z$, $k = \log_g h(m)$, $l = \log_g s$, $i = \log_g w$, $i \in Z_q$.

Then the equations $z = s^u A^v$ can be written as $g^j = g^{ul + va}$,

$w = h(m)^u g^v$ can be written as $g^i = g^{uk + v}$.

We have the linear congruences $lu + av = j \bmod q$ and $ku + v = i \bmod q$.

The determinant of this system is $ka - l$.

It is invertible ($(ka - l)^{-1}$ exists) mod q, since $s \neq h(m)^a$. Hence, the system has a unique solution $(u, v) \in Z_q \times Z_q$. (if $s = h(m)^a$, the determinant of this system is $kauv - kauv = 0$.)

endofproof2

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

4 Disavowal: Suppose that in a verification, the challenge $z = s^u A^v$ and the response w have been generated. That verification fails and the signer claims that the signature is wrong; that is, s is not the ath root of $h(m)$. We describe **how the signer can prove to the verifier that the signature is in fact incorrect**. *See next page*

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

	Signer		Verifier
Signature generation	$s = h(m)^a$		
Signature verification	$w = z^{a^{-1}}$ $= h(m)^{ub/a} g^v$	$\leftarrow z$ $w \rightarrow$	$u, v \in_R Z_q$, $z = s^u A^v$ $w =? h(m)^u g^v$ ***failure***
Signature disavowal (the 2nd verification)	$w' = z'^{a^{-1}}$ $= h(m)^{u'b/a} g^{v'}$	$\leftarrow z'$ $w' \rightarrow$	$u', v' \in_R Z_q$, $z' = s^{u'} A^{v'}$ $(w\, g^{-v})^{u'} =? (w'\, g^{-v'})^u$ The signature is indeed invalid if both sides are equal.

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

Lemma 12.7.2

If $s = h(m)^a$ and w is fixed with $w \neq h(m)^u g^v$, then for every $w' \in G$ there is exactly one pair $(u', v') \in Z_q \times Z_q$ such that $(w\, g^{-v})^{u'} = (w'\, g^{-v'})^u$.

Proof. Suppose that $s = h(m)^a$, ……..(1)

$$w \neq h(m)^u g^v, \text{ . ……..(2)}$$

$$\text{and } (w\, g^{-v})^{u'} = (w'\, g^{-v'})^u. \text{ …(3)}$$

For the value w' in eq. (3), we have $w' = h(m_1)^{u'} g^{v'}$. That is, w' is a correct response of a particular signature. From eq. (3), we have

$$(w\, g^{-v})^{u'} = (w'\, g^{-v'})^u = (h(m_1)^{u'} g^{v'}\, g^{-v'})^u = (h(m_1)^{u'})^u$$

$$(w\, g^{-v})^{u'} = h(m_1)^{u'u} \Rightarrow h(m_1)^u = (w\, g^{-v})$$

$$\Rightarrow w = h(m_1)^u g^v$$

12.7 Undeniable Signature

12.7.2 Chaum-van Antwerpen scheme

For the value w' in eq. (3), we have $w' = h(m_1)^{u'} g^{v'}$. That is, w' is a correct response of a particular signature. From eq. (3), we have

$$(w\, g^{-v})^{u'} = (w'\, g^{-v'})^{u} = (h(m_1)^{u'} g^{v'}\, g^{-v'})^{u} = (h(m_1)^{u'})^{u}$$

$$(w\, g^{-v})^{u'} = h(m_1)^{u'u} \Rightarrow h(m_1)^{u} = (w\, g^{-v})$$

$$\Rightarrow w = h(m_1)^{u}\, g^{v}$$

However, $s = h(m)^{a} = h(m_1)^{a}$, contradicts to eq. (2).

Therefore, if the signer cheating in the 1st phase of verification, he will pass the 2nd verification (disavowal signature) with probability $1/q$, by the Lemma 12.7.1.

The 2nd assertion of Lemma 12.7.1, if $s \neq h(m)^{a}$ and if $z \in G$, then for each $w \in G$ there is exactly one pair (u, v) such that $w = h(m)^{u}\, g^{v}$ and $z = s^{u}\, A^{v}$. *endofproof*

12.8 Blind Signature

When issuing a **blind signature**, the signer only knows that he signed something but he does not know what he signed.

Blind signatures were invented to construct **anonymous electronic payment systems**.