Cryptography

Cryptography

Cryptography is the science and study of secret writing.

Cryptanalysis is the science and study of methods of breaking ciphers.

Cryptology: cryptography and cryptanalysis.

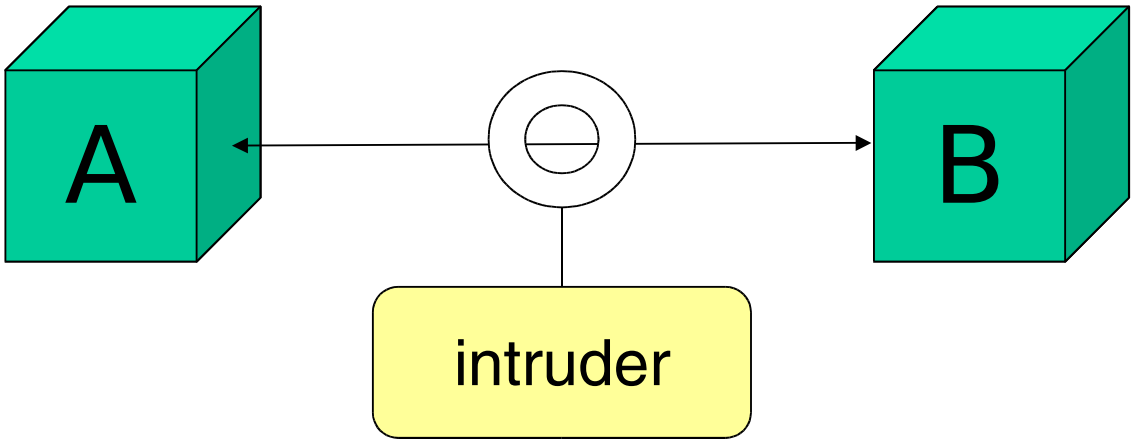
Today [HAC]: Cryptography is the study of mathematical techniques related to aspects of information security, such as confidentiality, data integrity, entity authentication, and data origin authentication.

The Origins of Cryptography

The enemy is an outsider listening to traffic

Two secure end systems communicate over an insecure channel

The Old Paradigm



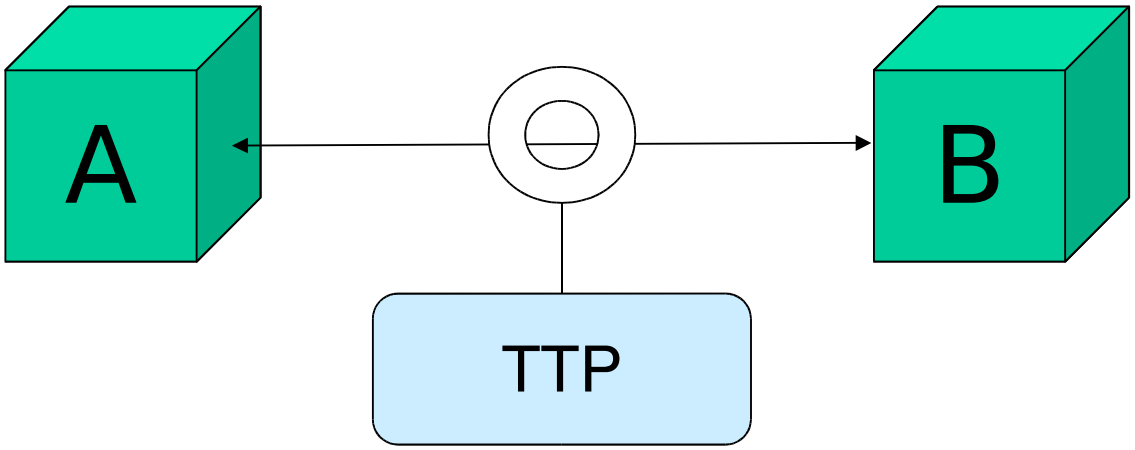
A and *B* communicate over an insecure channel.

A and *B* trust each other.

Intruder can read, delete, and insert messages.

With cryptography, *A* and *B* construct a secure logical channel over an insecure network.

The New Paradigm

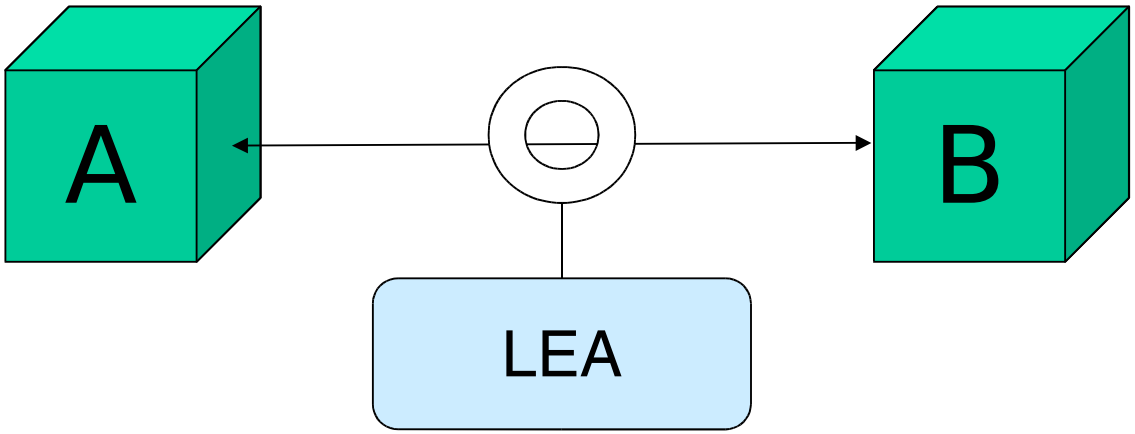


Electronic commerce: *A* and *B* are customer and merchant; they do not "trust" each other.

We want protection against insider fraud as much as protection against outsiders.

Trusted Third Parties help settle disputes.

Law Enforcement



In many countries laws regulate how a law enforcement agency (LEA) can intercept traffic.

Key recovery makes cryptographic keys available to their owner.

Key escrow makes keys available to a LEA.

Communications Security

Security services provided by cryptographic mechanisms:

- Data confidentiality: encryption algorithms hide the content of messages;
- Data integrity: integrity check functions provide the means to detect whether a document has been changed;
- Data origin authentication: message authentication codes or digital signature algorithms provide the means to verify the source and integrity of a message.

Data Integrity & Authentication

Data origin authentication includes data integrity: a message that has been modified in transit no longer comes from the original source.

Data integrity includes data origin authentication: when the sender's address is part of the message, you have to verify the source of a message when verifying its integrity.

Under the assumptions made, data integrity and data origin authentication are equivalent.

In other applications a separate notion of data integrity makes sense, e.g. for file protection in anti-virus software.

Modular Arithmetic

Basis for many modern cryptographic algorithms.

Let m be an integer (the modulus). Define an equivalence relation on the set of integers by $a \quad b \bmod m$ if and only if $a - b = \quad m$ for some integer .

We say "a is equivalent to b modulo m".

is an equivalence relation that divides the set of integers into m equivalence classes $(a)_m = \{ b \mid a \quad b \bmod m\}$, $0 \quad a < m$; we write $a \bmod m$ for $(a)_m$.

The following properties hold:

- $(a \bmod m) + (b \bmod m) \quad (a+b) \bmod m$,
- $(a \bmod m) \quad (b \bmod m) \quad (a \quad b) \bmod m$,
- for every $a \not\equiv 0 \bmod p$, p prime, there exists an integer a^{-1} so that $a \quad a^{-1} \quad 1 \bmod p$.

Multiplicative order modulo p: Let p be a prime and a an arbitrary integer; the multiplicative order of a modulo p is the smallest integer n so that $a^n \quad 1 \bmod p$.

Fermat's Little Theorem

Fermat's Little Theorem: for p prime and $a \quad 0 \mod p$, we have $a^{p-1} \quad 1 \mod p$.

Example: $p = 5$,

- $2^4 = 16 \quad 1 \mod 5$
- $3^4 = 81 \quad 1 \mod 5$
- $4^4 = 256 \quad 1 \mod 5$

Note: when computing $a^x \mod p$, you are working modulo $p-1$ in the exponent

Corollary for $n = p \quad q$, $e \quad d \quad 1 \mod lcm(p-1,q-1)$:

For a, $0 < a < n$, we have $a^{e \quad d} \quad a \mod n$.

Difficult Problems

Discrete Logarithm Problem (DLP): Given a prime modulus p, a basis a, and a value y, find the discrete logarithm of y, i.e. an integer x so that $y = a^x \bmod p$.

n-th Root Problem: Given integers m, n and a, find an integer b so that $a = b^n \bmod m$. The solution b is the n-th root of a modulo n.

Factorisation: Find the prime factors of an integer n.

With suitable parameters, these problems are a basis for many cryptographic algorithms.

However, not all instances of these problems are difficult to solve.

Integrity Check Functions

Integrity Protection – Example

To protect a program x, compute its hash $h(x)$ in a clean environment and store it in a place where it cannot be modified, e.g. on CD-ROM.

Protection of the hash value is important; computing the hash value requires no secret information, so anybody can create a valid hash for a given file.

To check whether the program has been modified, re-compute the hash value and compare it with the value stored.

One-way Functions

Requirements on a one-way function h:

Ease of computation: given x, it is easy to compute $h(x)$.

Compression: h maps inputs x of arbitrary bitlength to outputs $h(x)$ of a fixed bitlength n.

Pre-image resistance (one-way): given a value y, it is computationally infeasible to find an input x so that $h(x)=y$.

Collisions

The application just described needs more than the one-way property of h.

We are not concerned about an attacker reconstructing the program from the hash.

We are concerned about attackers who change program x to x' so that $h(x') = h(x)$.

Then, our integrity protection mechanism would fail to detect the change.

We say there is a collision when two inputs x and x' map to the same hash.

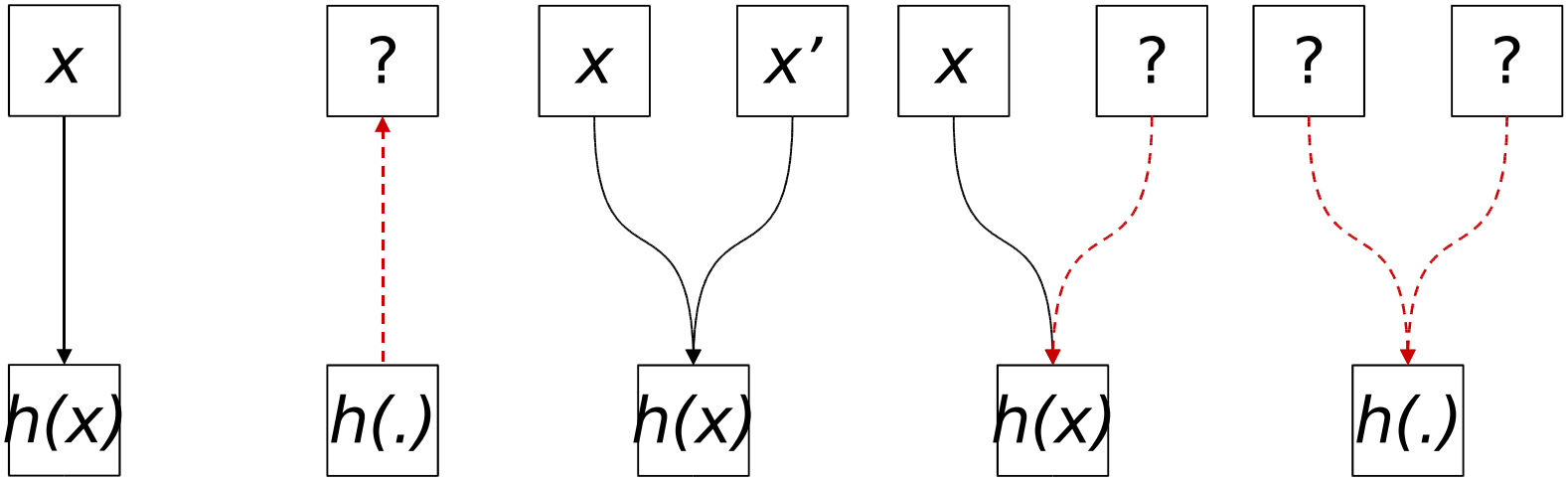
Collision Resistance

Integrity protection requires collision-resistant hash functions; we distinguish between:

2nd pre-image resistance (weak collision resistance): given an input x and $h(x)$, it is computationally infeasible to find another input x', $x \neq x'$, with $h(x)=h(x')$.

Collision resistance (strong collision resistance): it is computationally infeasible to find any two inputs x and x', $x \neq x'$, with $h(x)=h(x')$.

Properties of One-way Functions



ease of computation

pre-image resistance

collision

2nd pre-image resistance

collision resistance

Birthday Paradox

It depends on the bit-length of the hash how probable it is to find collisions by accident.

Given an n-bit hash y, the expected number of tries before an x with $h(x)=y$ is found is 2^{n-1}.

Given n-bit hash values, a set of $2^{n/2}$ inputs is likely to contain a pair causing a collision.

Birthday paradox: put m balls numbered 1 to m into an urn; draw a ball, list its number, and put it back; repeat; for m , the expected number of draws before a previously drawn number appears is sqrt($m/2$).

Manipulation Detection Codes

Manipulation detection code (MDC, also modification detection code, message integrity code): used to detect changes to a document.

Two types of MDCs:

- One-way hash function (OWHF): ease-of-computation, compression, pre-image resistance, and 2nd pre-image resistance.
- Collision resistant hash function (CRHF): compression, ease-of-computation, 2nd pre-image resistance, and collision resistance.

Checksums

The result of applying a hash function is called hash value, message digest, or checksum.

The last term creates frequent confusion .

In communications, checksums often refer to error correcting codes, typically a cyclic redundancy check (CRC).

Checksums used by anti-virus products, on the other hand, must not be computed with a CRC but with a cryptographic hash function.

Discrete Exponentiation

Discrete exponentiation: $h(x) := g^x \bmod p$.

Discrete Logarithm Problem (DLP): given y find the "logarithm" x so that $y = g^x \bmod p$.

For a judicious choices of parameters p and g the DLP is difficult to solve and discrete exponentiation is a one-way function.

Discrete exponentiation is a useful primitive in the construction of cryptographic schemes but it is too slow for many applications.

Construction

Pattern for the design of fast hash functions:

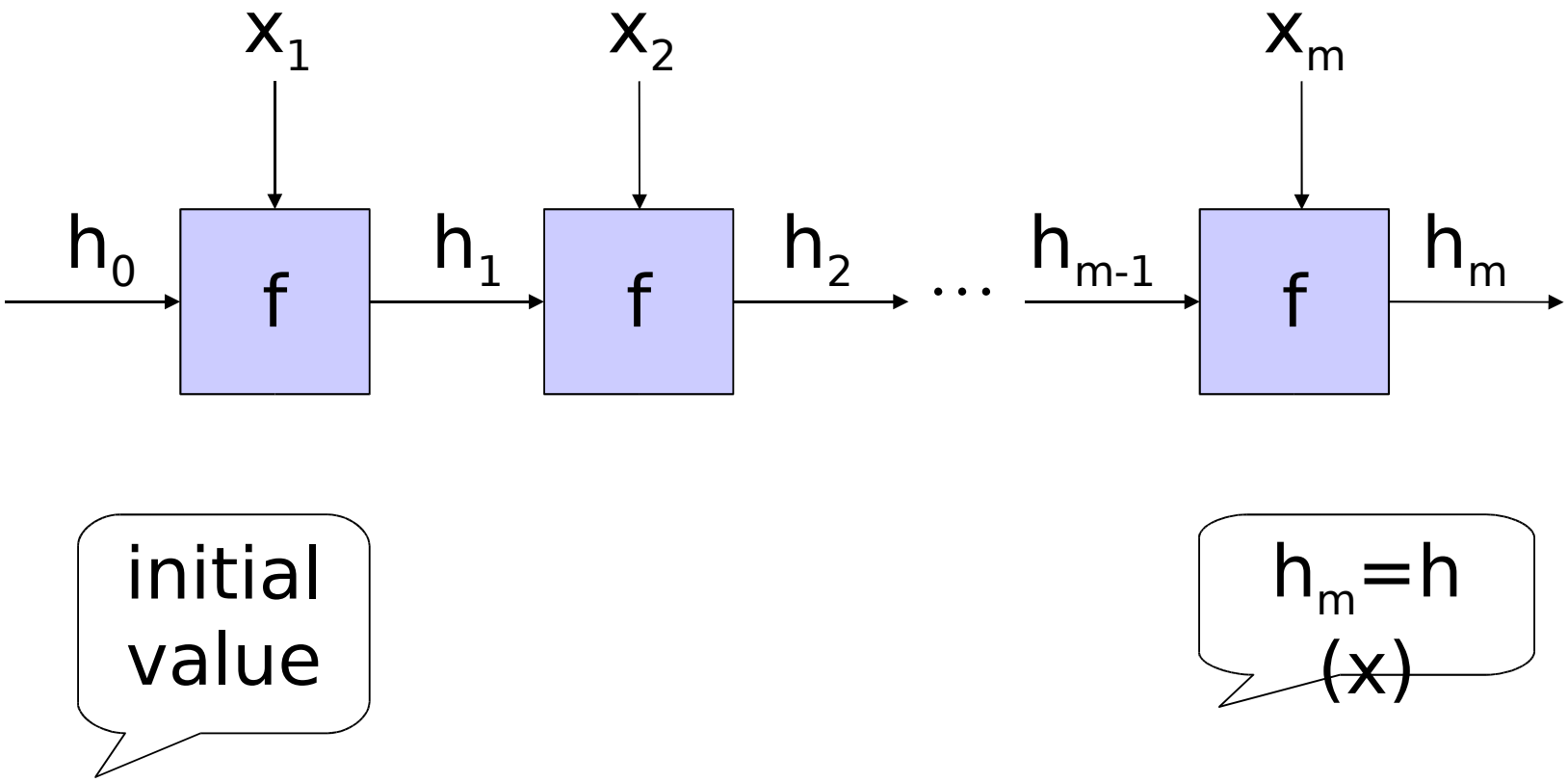
The core of the hash function is a compression function f that works on fixed size input blocks.

An input x of arbitrary length is broken up into blocks $x_1,\ldots, x_m$ of the given block size; the last block has to be padded.

Compute the hash of x by repeatedly applying the compression function: with a (fixed) initial value h_0, compute $h_i = f(x_i || h_{i-1})$ for $i=1,\ldots, m$ and take h_m as the hash value of x.

The symbol || denotes concatenation.

Construction



Frequently Used Hash Functions

MD4: weak, it is computationally feasible to find meaningful collisions.

MD5: standard choice in Internet protocols but similar in design to MD4 and no longer recommended.

Secure Hash Algorithm (SHA-1): designed to operate with the US Digital Signature Standard (DSA); 160-bit hash value.

RIPEMD-160: hash function frequently used by European cryptographic service providers.

News on Hash Functions

News (early 2005): "SHA-1 has been broken."

No details available yet; previous cryptanalysis of hash functions had found methods for constructing pairs of inputs that map to the same hash value.

Reports that collisions for SHA-1 can be found in 269 "steps".

For 160-bit hash values, the yardstick is the computation of 2^{80} random hash values.

Longer hash values are advisable: SHA-256

Message Authentication Codes

In communications, we should not rely on secure storage to protect hash values.

Use secrets instead: compute a MAC $h_k(x)$ from the message x and a secret key k.

To authenticate a message, the receiver has to share the secret key used to compute the MAC with the sender.

A third party that does not know the key cannot validate the MAC.

Message Authentication Codes

A MAC must have the compression and ease-of-computation property, and an additional computation resistance property:

For any fixed value of k unknown to the adversary, given a set of values $(x_i, h_k(x_i))$, it is computationally infeasible to compute $h_k(x)$ for any new input x.

Message Authentication Codes (keyed hash functions) support data origin authentication services.

HMAC

A MAC algorithm can be derived from a MDC algorithm h using the HMAC construction:

For a given key k and message x, compute

$$HMAC(x) = h(k||p_1||h(k||p_2||x))$$

where p_1 and p_2 are bit strings (padding) that extend k to a full block length of the compression function used in h.

HMAC is specified in Internet RFC 2104.

Digital Signatures

Digital Signature Mechanisms

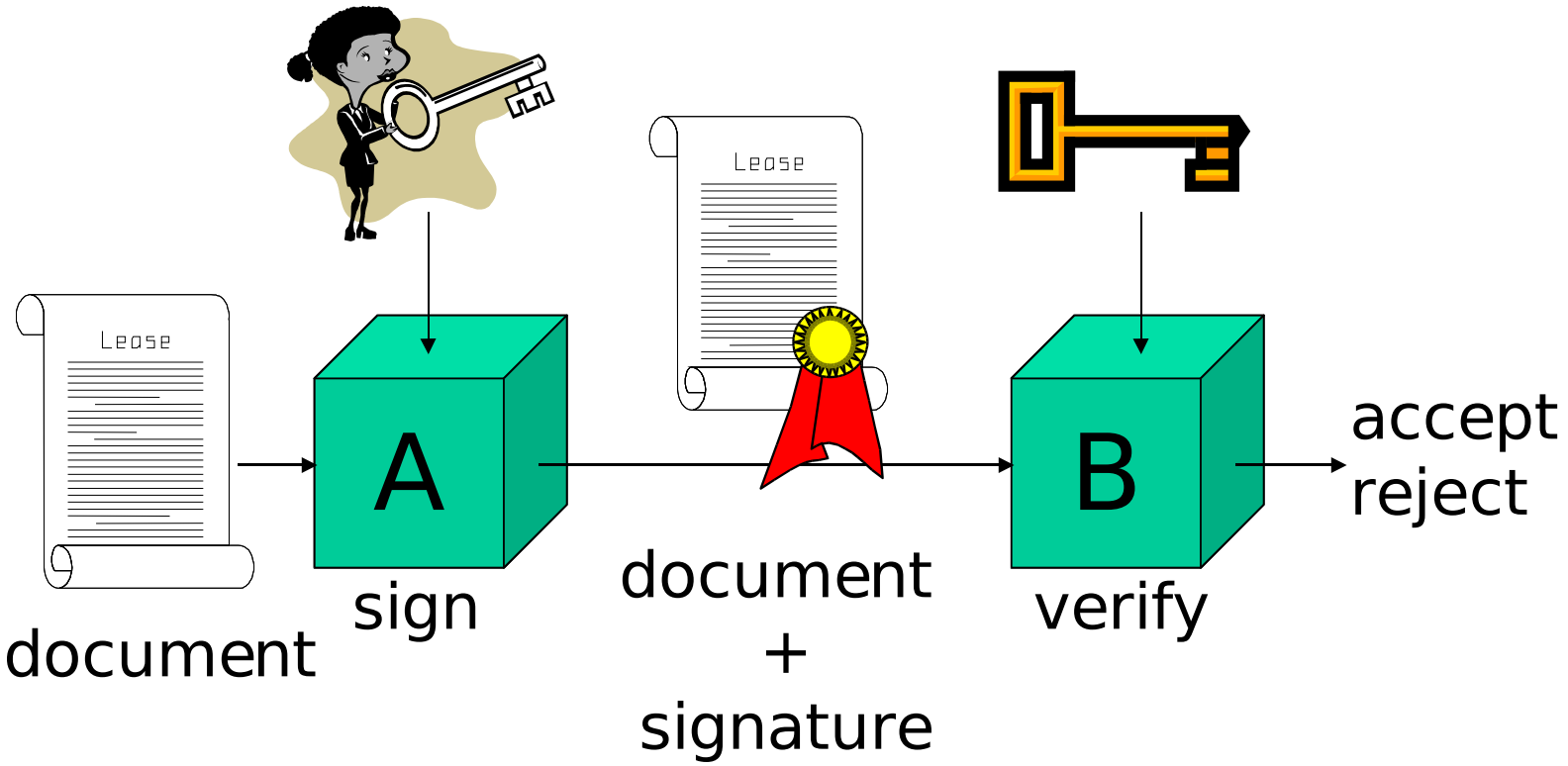
A MAC cannot be used as evidence that should be verified by a third party.

Digital signatures used for non-repudiation, data origin authentication and data integrity services, and in some authentication exchange mechanisms.

Digital signature mechanisms have three components:

- key generation
- signing procedure (private)
- verification procedure (public)

Digital Signatures



Digital Signatures

A has a public verification key and a private signature key (public key cryptography).

A uses her private key to compute her signature on document *m*.

B uses a public verification key to check the signature on a document *m* he receives.

At this technical level, digital signatures are a cryptographic mechanism for associating documents with verification keys.

Digital Signatures

To get an authentication service that links a document to *A*'s name (identity) and not just a verification key, we require a procedure for *B* to get an authentic copy of *A*'s public key.

Only then do we have a service that proves the authenticity of documents 'signed by *A*'.

Yet even such a service does not provide non-repudiation at the level of persons.

One-time Signatures

Make use of a a cryptographic hash function h.

Key generation: to sign an n-bit document, pick your private key by choosing at random $2n$ values x_{i0}, x_{i1}; publish commitments $y_{i0} = h(x_{i0})$, $y_{i1} = h(x_{i1})$, $1 \quad I \quad n$, as your public key.

Signing: the i-th bit of the signature s of document m is given by $s_i = x_{i0}$ if $m_i = 0$, $s_i = x_{i1}$ if $m_i = 1$.

The private key can be used once only.

Verification: the verifier has the public key and checks whether $y_{i0} = h(s_{i,})$ if $m_i = 0$ and whether $y_{i1} = h(s_{i,})$ if $m_i = 1$.

The verifier needs additional evidence to confirm that the values y_{i0}, y_{i1} are indeed your public key.

RSA Signatures

The RSA (Rivest, Shamir, Adleman) algorithm can be used for signing and for encryption.

This property peculiar to RSA has led to many misconceptions about digital signatures and public key cryptography.

Key generation:

- user A picks two prime numbers p, q.
- Private signature key: an integer d with $gcd(d,p-1) = 1$ and $gcd(d,q-1) = 1$.
- Public verification key: $n = p \ \ q$ and an integer e with $e \ \ d \quad 1 \bmod lcm(p-1,q-1)$.

Factorization & RSA

Factorization: given an integer n, find its prime factors.

Finding small factors is “easy”

Testing for primality is “easy”.

Factoring an RSA modulus $n = p \ \ q$ is ‘difficult’.

When the public modulus $n = p \ \ q$ can be factored, the security of RSA is compromised.

There exists no proof that the security of RSA is equivalent to the difficulty of factoring.

RSA Signatures

Signing: the signer A hashes the document m so that $0 < h(m) < n$ and computes the signature $s = h(m)^d \bmod n$.

Verification: the verifier uses a verification key (n,e) and checks $s^e \quad h(m) \bmod n$.

For a correct signature, this equation holds because $s^e = h(m)^{d\ e} = h(m) \bmod n$.

The hash function adds an important redundancy check to signature verification.

RSA Signatures

If signature verification does not include a redundancy check, existential forgeries are possible.

In RSA the public verification key can be chosen so that signature verification is particularly quick, e.g. $e=2^{16}+1$.

Signatures with message recovery: there is a mode of RSA where short documents can be recovered from the signature and do not have to be transmitted separately.

Digital Signature Algorithm

Key generation:

- Select a prime q such that $2^{159} < q < 2^{160}$.
- Select an integer t, $0 \le t \le 8$, and a prime p, $2^{511+64t} < p < 2^{512+64t}$, so that q divides $p-1$.
- Select , $1 <$ $< p-1$, and compute $g =$ $^{(p-1)/q} \bmod p$; if $g = 1$, try again with a new .
- A selects a, $1 \le a \le q-1$, and computes $y = g^a \bmod p$.
- A's private key is a, the public key is (p,q,g,y).

DSA uses SHA-1 as its hash functions; hash values $h(m)$ are converted into integers.

ECDSA: similar to DSA, based on elliptic curves.

Digital Signature Algorithm

Signature generation

Input: private key a, public values g,p,q, message hash $h(m)$

Select k at random, $0<k<q$

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

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

signature: (r,s)

Signature verification

Input: signature (r,s), public values p,q,g, $y=g^a \bmod p$, hash $h(m)$

verify $0<r<q$, $0<s<q$

$w = s^{-1} \bmod q$

$u1 = w \cdot h(m) \bmod q$ $\quad u2 = r \cdot w \bmod q$

$v = (g^{u1} y^{u2} \bmod p) \bmod q$

Accept if and only if $v=r$

MACs & Digital Signatures

MACs and digital signatures are authentication mechanisms.

MAC: the verifier needs the secret that was used to compute the MAC; thus a MAC is unsuitable as evidence with a third party.

- The third party would need the secret.
- The third party cannot distinguish between the parties knowing the secret.

In contrast, digital signatures can be used as evidence with a third party.

MACs & Digital Signatures

MACs are sometimes called “signatures”; this can create wrong expectations and should be avoided.

The term “non-repudiation” was coined to distinguish the features of authentication based on digital signatures from MAC-based authentication.

At this level, non-repudiation can be given a precise technical meaning.

Encryption

Terminology

Encryption: plaintext (clear text) x is converted into a ciphertext under the control of a key K.

- We write $eK(x)$.

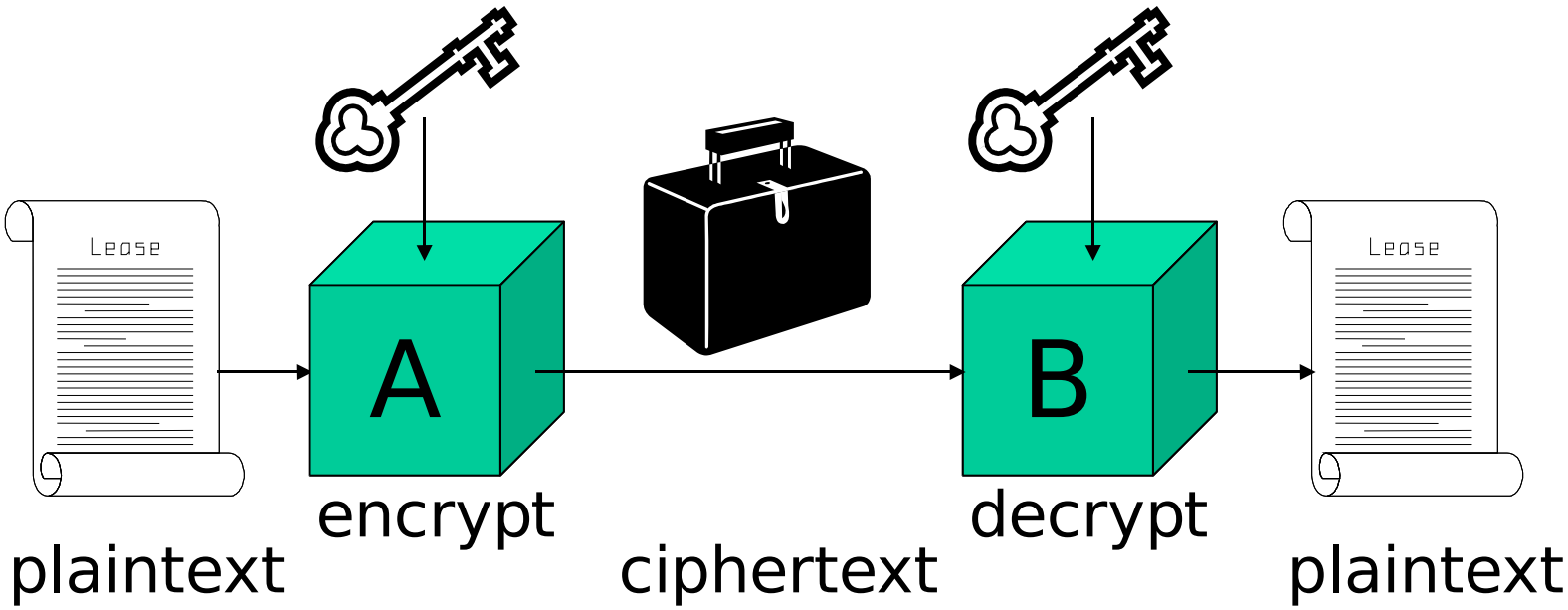
Decryption with key K computes the plaintext from the ciphertext y.

- We write $dK(y)$.

Symmetric ciphers: the decryption key is essentially the same as the encryption key.

Asymmetric ciphers: it is computationally infeasible to derive the private decryption key from the corresponding public encryption key.

Symmetric Key Encryption



Symmetric Key Cryptography

Protects documents on the way from A to B.

A and B need to share a key.

A and B have to keep their keys secret (secret key cryptography).

There has to be a procedure whereby A and B can obtain their shared key.

For n parties to communicate directly, about n^2 keys are needed.

Block Ciphers & Stream Ciphers

Block ciphers: encrypt sequences of "long" data blocks without changing the key.

- Security relies on design of encryption function.
- Typical block length: 64 bits, 128 bits.

Stream ciphers: encrypt sequences of "short" data blocks under a changing key stream.

- Security relies on design of key stream generator.
- Encryption can be quite simple, e.g. XOR.
- Typical block length: 1 bit, 1 byte, 8-bit word.

Block Cipher Basics

Given a block cipher with n-bit blocks, for any key K the function $eK(x)$ is a permutation on the set of n-bit blocks.

Each key defines a different permutation.

For each permutation, observing a ciphertext block should not increase the information about the corresponding plaintext block.

For any given plaintext block x, encryption $eK(x)$ should change about half of the bits.

Substitution and Permutation

It should be computationally difficult to compute keys from plaintext/ciphertext pairs.

This is only sketches the requirements on a block cipher; for precise requirements please refer to the literature.

Basic principles in block cipher design:

- Substitution: replace bit patterns in the input so that the output bits are not indicative of the input.
- Permutation: change bit positions so that changing an input bit affects a different bit of the output.

Substitution – Example

input bits 2 – 5

6-to-4 bit substitution box from DES

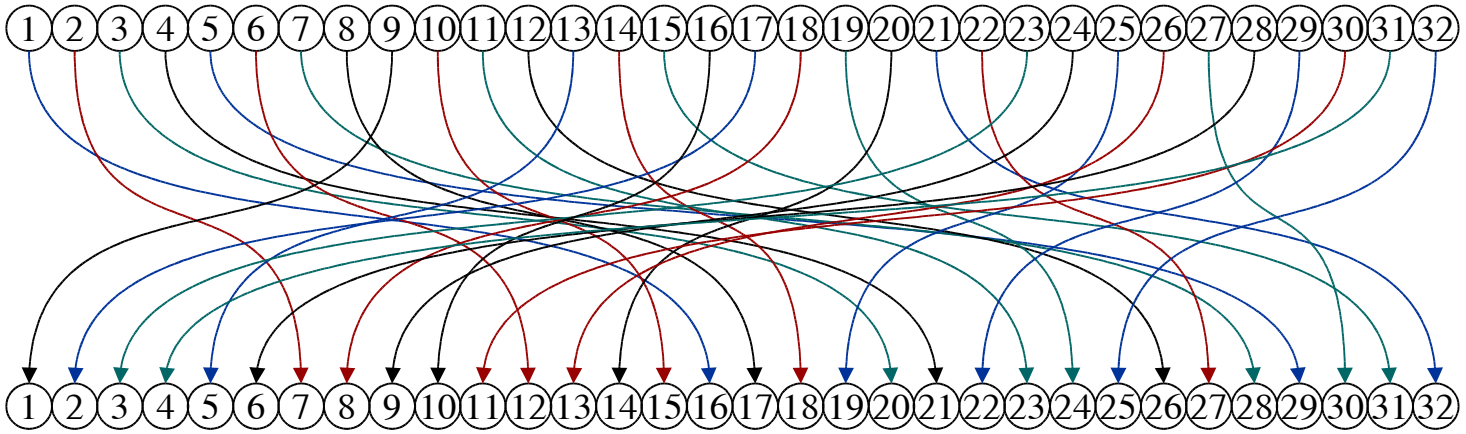
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

input bits 1 and 6

Permutation – Example

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

permutation of 32 bits



Round Structure

To facilitate efficient encryption & decryption, block ciphers usually have a round structure.

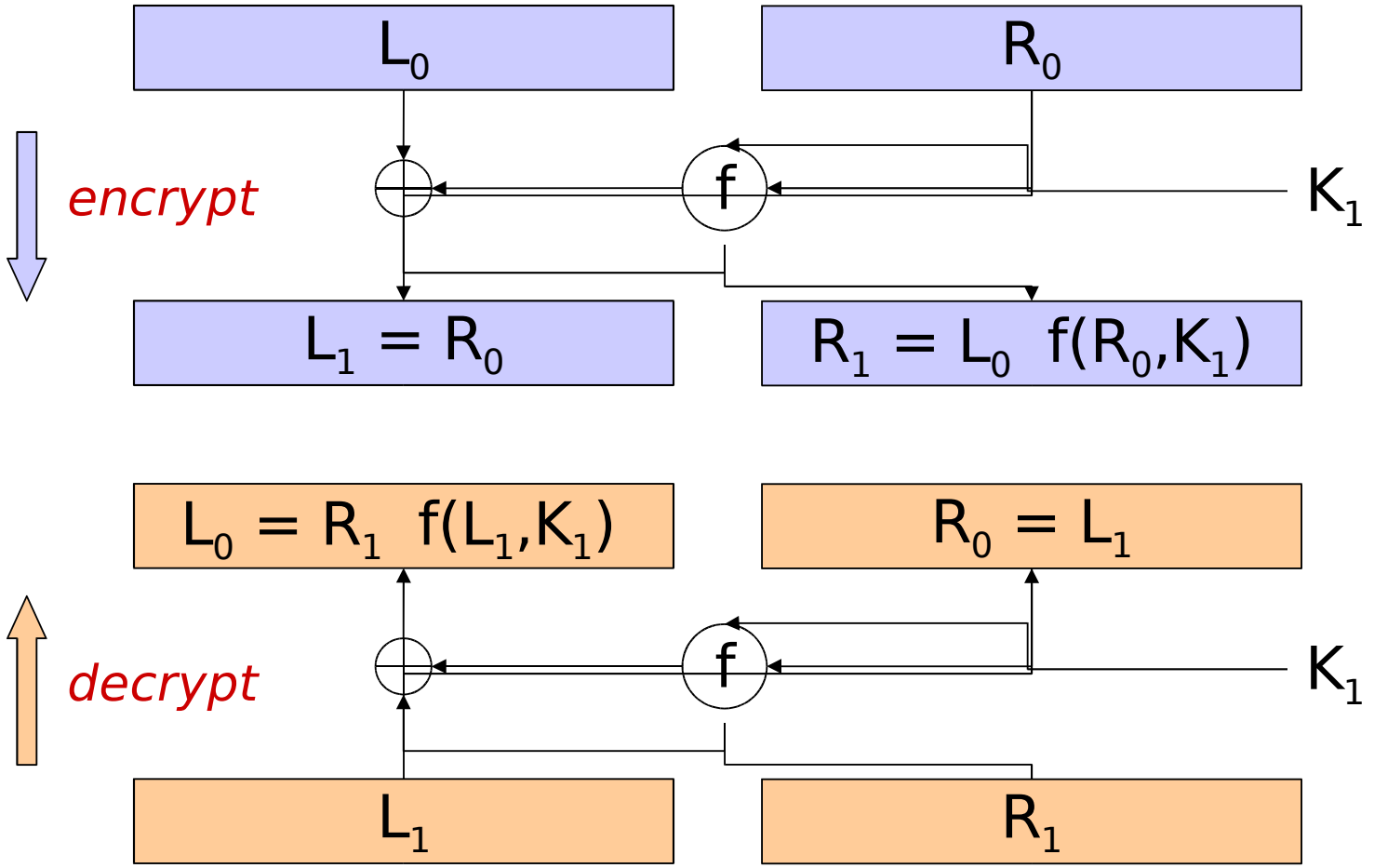
Each round depends on a sub-key; each round in itself is not very “secure”.

Security through iteration:

- How many rounds do you want?

We would like to use the same structures for encryption and decryption.

Feistel Ciphers



Algorithms

DES (more in a moment)

AES (more in a moment)

Triple-DES: ANSI X9.45, ISO 8372

FEAL

IDEA

SAFER

Blowfish, Mars, Serpent, …

and many more

Data Encryption Standard

Published in 1977 by the US National Bureau of Standards for use in unclassified government applications with a 15 year life time.

Feistel cipher with 64-bit data blocks, 56-bit keys.

56-bit keys were controversial in 1977; today, exhaustive search on 56-bit keys is very feasible.

Controversial because of classified design criteria, however no loop hole has yet emerged.

DES designed to resist differential cryptanalysis.

Advanced Encryption Standard

Public competition to replace DES: 56-bit keys and 64-bit data blocks no longer adequate.

Rijndael nominated as the new Advanced Encryption Standard (AES) in 2001 [FIPS-197].

Rijndael (pronounce as “Rhine-doll”) designed by Vincent Rijmen and Joan Daemen.

Versions for 128-bit, 196-bit, and 256-bit data and key blocks (all combinations of block length and key length are possible).

Rijndael is not a Feistel cipher.

Comments on Security

Single DES should no longer be used, triple DES used in the financial sector.

Recommended key length: 80-90 bits.

No provable security.

Algorithms designed to resist known attacks: e.g. differential & linear cryptanalysis.

It is not recommended to design your own algorithms; amateur designs are usually broken quite easily.

Using Encryption for Real

With a block cipher, encrypting a n-bit block x with a key K gives a ciphertext block $y = eK(x)$.

Given a well designed block cipher, observing y would tell an adversary nothing about x or K.

What happens if the adversary observes traffic over a longer period of time?

- The adversary can detect if the same message had been sent before; if there are only two likely messages "buy" and "sell" it may be possible to guess the plaintext without breaking the cipher.

Electronic Code Book Mode

Electronic code book (ECB): data blocks are encrypted independently under the same key.

Even when an algorithm is “secure” with respect to single blocks, ciphertexts still leak information about the structure of messages consisting of a sequence of blocks.

We prefer to use block ciphers in modes that map different encryptions of the same plaintext to different ciphertexts.

Error Propagation

The Hamming distance $d(x,x')$ between two blocks x, x' is the number of positions where x and x' differ.

With a single bit error when transmitting y, a ciphertext block y' is received with $d(y,y') = 1$.

For a n-bit block cipher we should expect $d(x,x') \approx n/2$ for the decryption result $x' = dK(y')$.

A high error propagation rate is a desirable security feature but a disadvantage when sending encrypted data over noisy channels.

Cipher Block Chaining mode

Cipher block chaining (CBC): cipher block C_i depends on the previous block C_{i-1}.

$$C_i = eK(P_i \quad C_{i-1}) \qquad \text{(encrypt)}$$

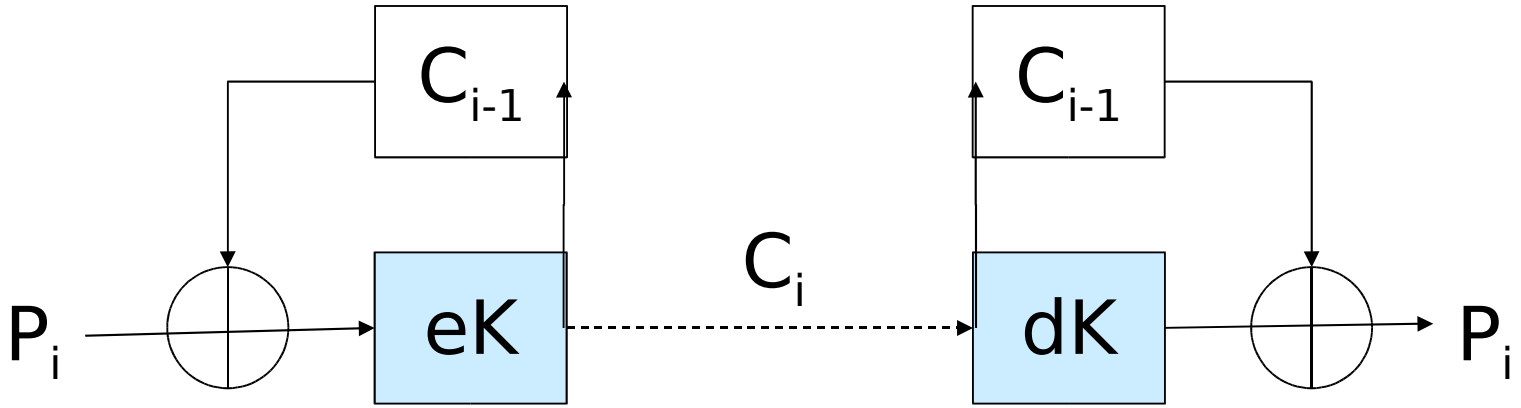
$$P_i = C_{i-1} \quad dK(C_i) \quad \text{(decrypt)}$$

Note: $C_{i-1} \quad dK(C_i) = C_{i-1} \quad P_i \quad C_{i-1} = P_i$.

For processing the first block, an initialization vector (IV) C_0 is used.

The integrity of C_0 has to be protected; it is usually kept secret .

Cipher Block Chaining mode



A corrupted ciphertext block C_i affects only the two plaintext blocks P_i, P_{i+1}.

Cipher Block Chaining mode

Different encryptions of the same plaintext give the same ciphertext.

Repeated plaintext blocks do not show up as repeated blocks in the ciphertext.

Reordering of ciphertext blocks leads to decryption errors.

As CBC creates a link between plaintext blocks, it is being used as a basis for constructing message authentication codes.

Output Feedback Mode

Output feedback (OFB): k-bit key K, n-bit IV, r-bit plaintext blocks.

Block cipher used as a key stream generator

Internal variable S; $S_0 = \text{IV}$, $S_i = eK(S_{i-1})$.

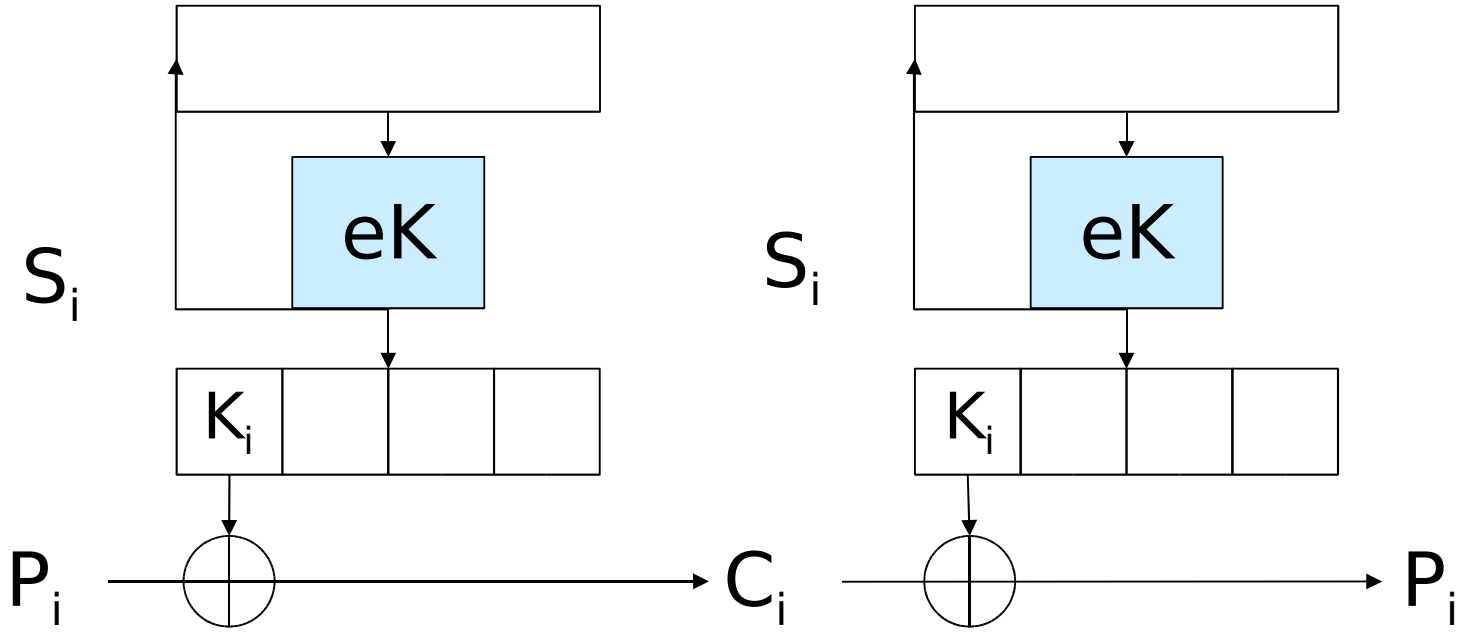
The key K_i for processing the i-th block: the r leftmost bits of S_i; encryption & decryption:

$$C_i = P_i \quad K_i \qquad \text{(encrypt)}$$

$$P_i = C_i \quad K_i \qquad \text{(decrypt)}$$

The IV need not be kept secret.

Output Feedback Mode (OFB)



A bit error in the ciphertext affects exactly the same bit in the plaintext.

Output Feedback Mode (OFB)

Repeated plaintext blocks do not show up as repeated blocks in the ciphertext.

Different encryptions of the same plaintext with the same key and IV give the same ciphertext.

Encryption of different plaintexts with the same key and IV reveals information about the plaintexts: if $C_i = P_i \quad K_i$ and $C'_i = P'_i \quad K_i$ then

$$C_i \quad C'_i = P_i \quad K_i \quad P'_i \quad K_i = P_i \quad P'_i .$$

A Note on Plaintexts

If plaintexts are natural language documents or other structured documents, plaintext blocks will not be randomly distributed.

If the distribution of plaintext blocks is known we can compute the distribution of $P_i \quad P'_i$ and reconstruct P_i and P'_i by statistical means.

Once we have a plaintext, we can compute the key $K_i = P_i \quad C_i$.

Cipher Feedback Mode

Cipher feedback (CFB): k-bit key K, n-bit IV, r-bit data blocks; IV need not be kept secret.

Block cipher used as a data dependent key stream generator.

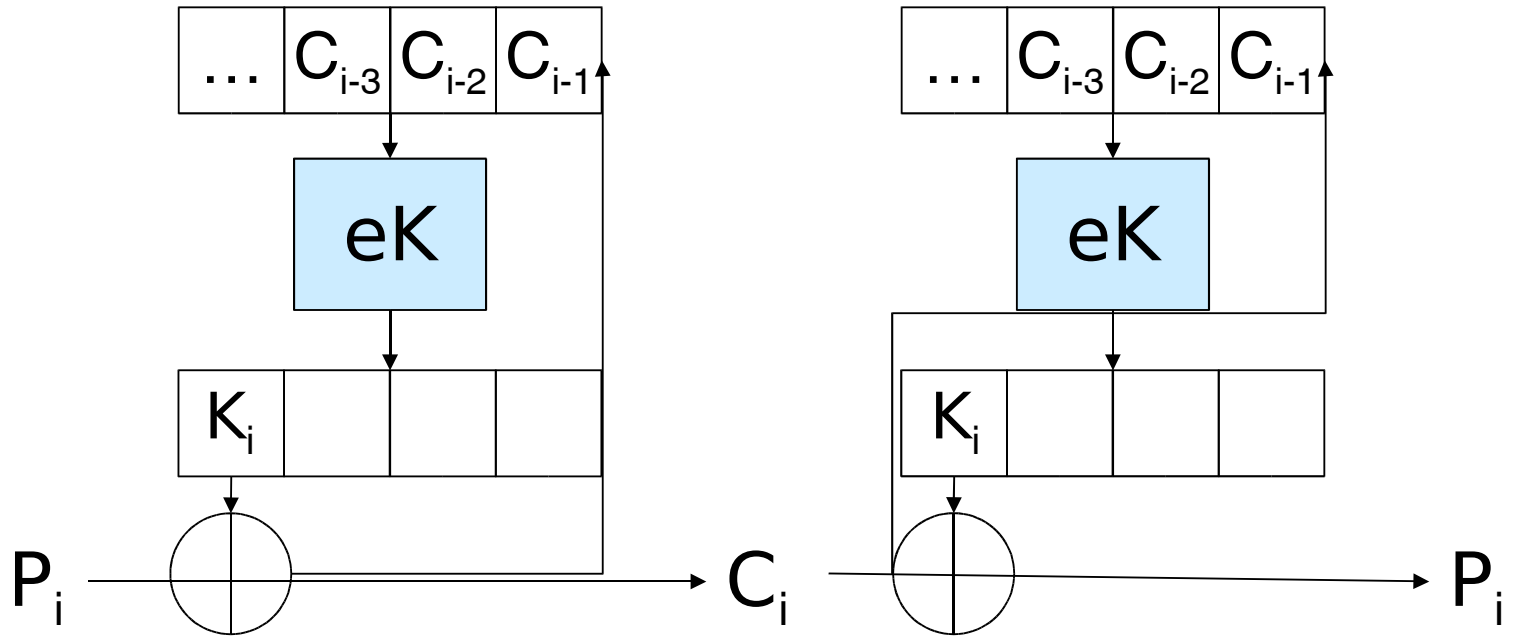
Internal variable S; S_0=IV, $S_i = 2^r S_{i-1} + C_i \bmod 2^n$

Key K_i for processing the i-th block: r leftmost bits of $e_K(S_i)$; encryption & decryption:

$$C_i = P_i \quad K_i \qquad \text{(encrypt)}$$

$$P_i = C_i \quad K_i \qquad \text{(decrypt)}$$

Cipher Feedback Mode (CFB)



Cipher Feedback Mode (CFB)

Repeated plaintext blocks do not show up as repeated blocks in the ciphertext.

Different encryptions of the same plaintext with the same key and IV give the same ciphertext.

Encryption of different plaintexts with the same key and IV is not a security problem.

A single bit error in a ciphertext block affects decryption until this block is shifted out of the register of the key generator.

More Modes

CTR (confidentiality) mode: counter mode.

OMAC (authentication) mode: One Key CBC MAC mode.

CCMB (authentication and encryption) mode: counter with CBC-MAC mode, developed for WLAN (IEEE 802.11i).

- NIST Computer Security Resource Center: draft special publications 800-38B, 800-38C

Galois Counter Mode (GCM)

Carter-Wegman + Counter (CWC) mode

Stream Ciphers

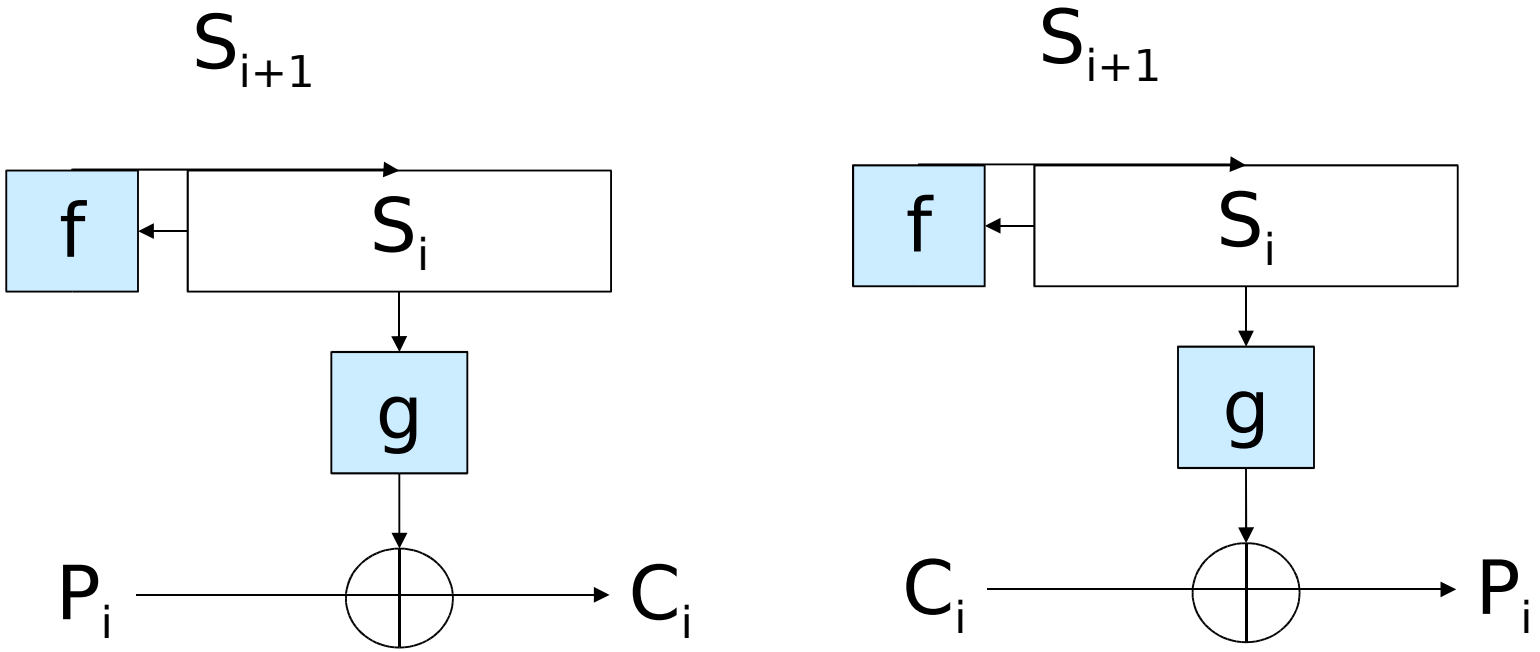
Consist of a key stream generator and a function for combining key stream and data.

The combing function tends to be simple, exclusive-or is a typical example.

The key stream generator takes as its input a seed S_0 and updates its state with a state transition function f, $S_{i+1} = f(S_i)$.

The output at step i is a key K_i derived from S_i as $K_i = g(S_i)$.

Stream Ciphers



Encryption and decryption are usually identical operations.

Stream Ciphers

In such a cipher, a bit error in ciphertext bit *i* causes a single bit error in plaintext bit *i*.

Wireless networks use stream ciphers to protect data confidentiality.

An adversary can make precise relative changes to the plaintext by modifying the corresponding ciphertext bits.

Stream ciphers therefore cannot be used for integrity protection.

Public Key Encryption

Proposed in the open literature by Diffie & Hellman in 1976.

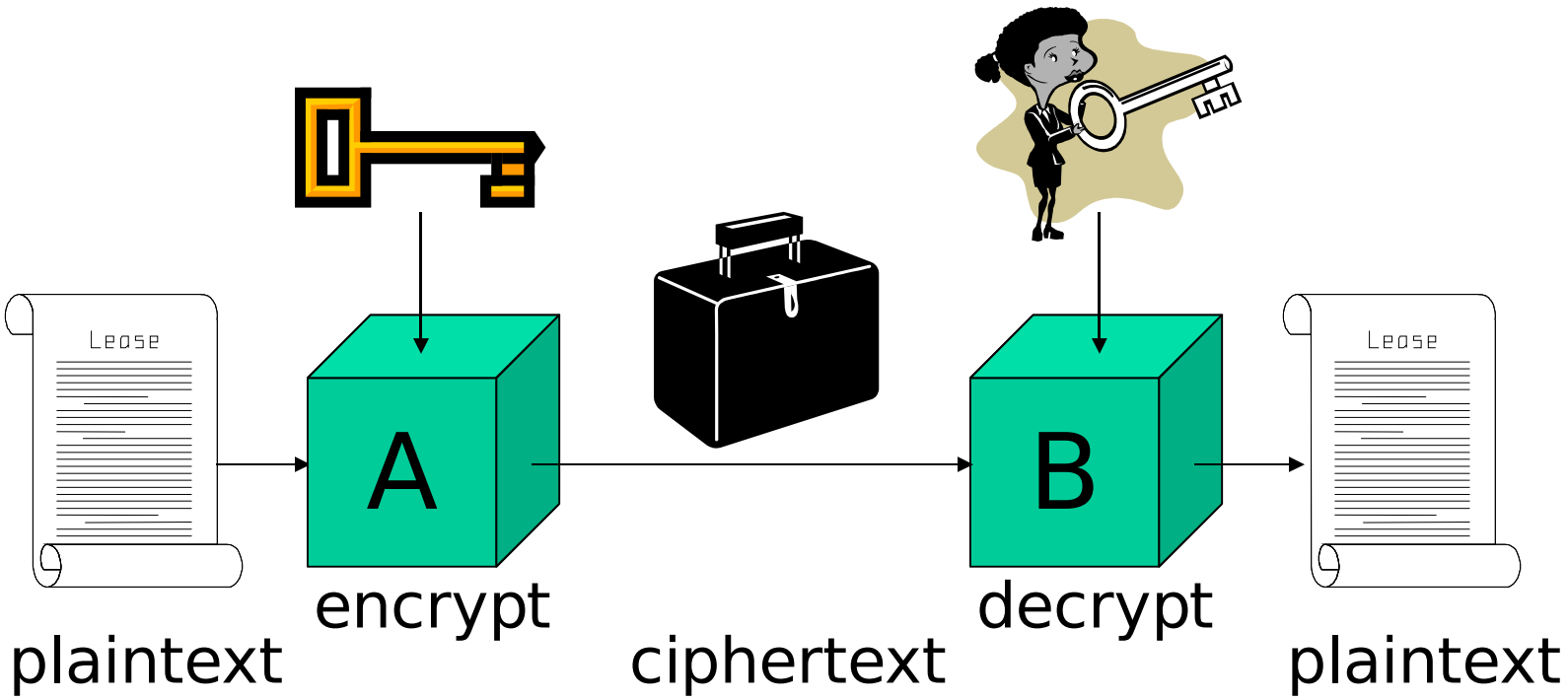
Each party has a public encryption key and a private decryption key.

Computing the private key from the public key should be computationally infeasible.

The public key need not be kept secret but it is not necessarily known to everyone.

There exist applications where access to public keys is restricted.

Encryption with Public Keys



Public Key Encryption

Encryption protects documents on the way from *A* to *B*.

B has a public encryption key and a private decryption key.

A procedure is required for *A* to get an authentic copy of *B*'s public key (need not be easier than getting a shared secret key).

For *n* parties to communicate, *n* key pairs are needed.

Public Key Infrastructures

"With public key cryptography, you can send messages securely to a stranger".

This is not really true; how do you know who has got the private key corresponding to the public key you are using?

How do you get a public key for a party you want to send a message to?

Additional "public key infrastructures" are needed to link persons to keys.

RSA Encryption

We have already discussed the RSA (Rivest, Shamir, Adleman) signature algorithm.

The RSA encryption algorithm is based on the same principles.

Key generation:

- User A picks two prime numbers p, q.
- Public encryption key: $n = p \quad q$ and an integer e with $gcd(e,p-1) = 1$ and $gcd(e,q-1) = 1$.
- Private decryption key: an integer d with $e \quad d \quad 1 \bmod lcm(p-1,q-1)$.

RSA Encryption

Messages are broken into message blocks m_i of length $0 < m_i < n$.

Encryption: sender A takes a message block m and computes the ciphertext $c = m^e \bmod n$.

Decryption: the receiver uses its decryption exponent d and computes $m = c^d \bmod n$.

Note: $c^d = m^{e\ d} = m \bmod n$.

Don't be deceived by the simplicity of RSA, proper implementation can be quite tricky.

Padding

RSA is a block cipher; keys are chosen so that the block length is 1024 bit (or 2048, 4096, …)

When encrypting a message, padding may have to be added to make the message length a multiple of the block length.

Padding can defeat some attacks: when decrypting a message, the receiver can check the padding data and discard plaintexts with syntactically incorrect padding.

Padding as Source of Attacks

PKCS #1 v1.5 encoding of a data value *D*:

- 00, 02: bytes with values 0 and 2 respectively
- *PS*: string of pseudo randomly generated non-zero bytes of length $|n| - |D| - 3$ (|.| gives length in bytes)

Bleichenbacher's attack: Uses 2^{20} chosen ciphertexts to get the plaintext if the receiver signals whether decryption fails or succeeds.

Bleichenbacher's Attack

Typical setting (SSL): Data value is a session key, the receiver is a server.

Attacker intercepts an encrypted session key.

Attacker sends a chosen ciphertext to server.

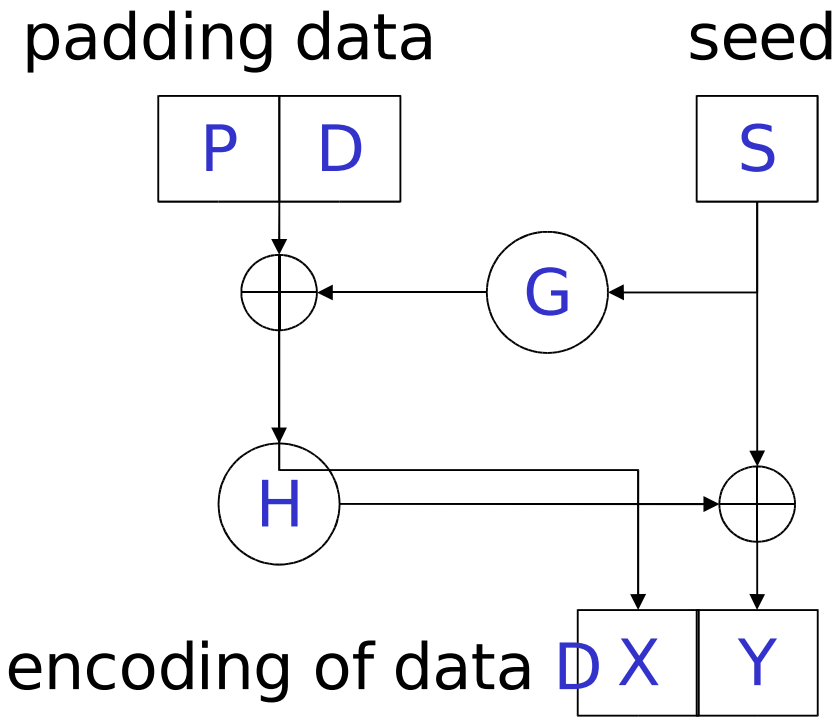
Server replies with an error message when decryption fails.

No error signals success and narrows the interval containing the session key.

After 2^{20} attempts the key is uniquely defined (in crypto, 1 000 000 can be a small number).

OAEP (PKCS #1 v2.1)

Optimal Asymmetric Encryption Padding (simplified)



Reconstructing D:

S H(X) Y

P||D X G(S)

(|| … concatenation)

OAEP (PKCS #1 v2.1)

S is a randomly generated seed.

P is some padding.

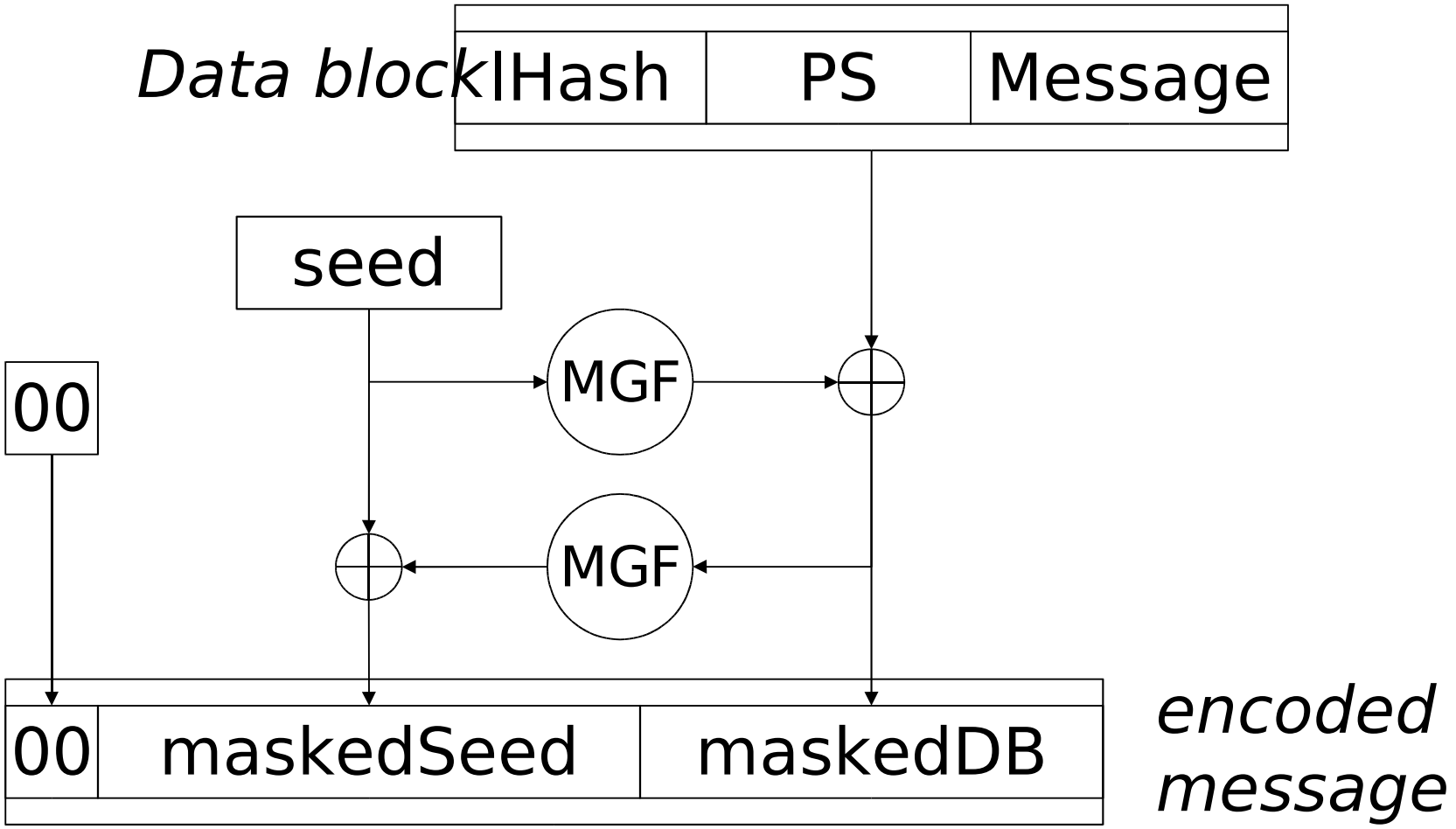
G, *H* are mask generation functions (MGFs) (typically based on SHA-1).

D is easily derived from its encoding.

Difficult to predict anything nontrivial about the encoding from *D* without knowing *S*.

The OAEP construction is underpinned by a security proof.

OAEP version 2.1



More on OAEP

OAEP came with a security proof, but the proof was wrong.

The proof was fixed, but at the expense of somewhat weaker results.

- Some bounds in the proof had to be relaxed.

New attacks against OAEP (Manger, Crypto 2001).

Research and standardization of padding methods is an ongoing effort.

Strength of Mechanisms

Measuring the strength of cryptographic algorithms is an imprecise art.

Empirical security: an algorithm has withstood the test of time.

Provable security: an algorithm is provably secure if breaking the algorithm is at least as difficult as solving some hard problem.

- 'At least as difficult' is an asymptotic concept.
- We don't know whether factorization or DLP are really hard.

Unconditionally security: secure against attackers with unlimited computing power.

Performance

algorithm	cycles/byte
RC4	7-8
MD5	7-8
SHA-1	15
SHA-512	83
Rijndael-128	25-30
DES	60

Based on data from NESSIE project

Performance

algorithm	operation	cycles per invocation	key setup (cycles)	key length (bits)
RSA-OAEP	encrypt	2.026 M	1.654 M	1024
	decrypt	42.000 M		
RSA-PSS	sign	42.000 M	1.334 M	1024
	verify	2.029 M		
ECDSA GF(p)	sign	4.775 M	4.669 M	160
	verify	6.085 M		
ECDSA GF(2^m)	sign	5.061 M	4.825 M	163
	verify	6.809 M		