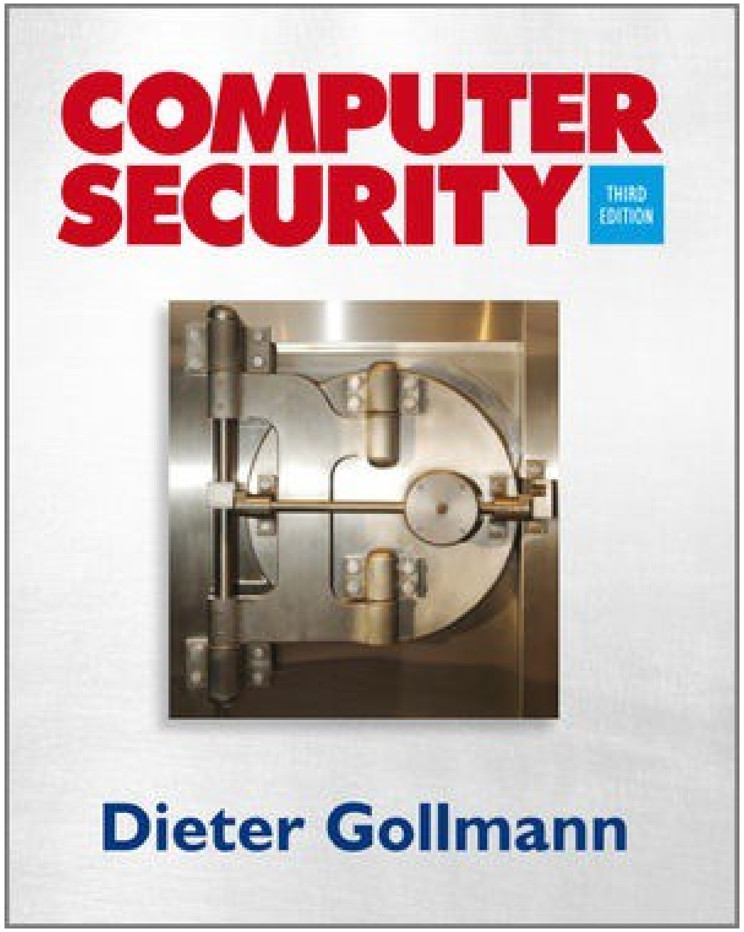
Computer Security 3e

Dieter Gollmann



Chapter 1: Cryptography

Cryptography

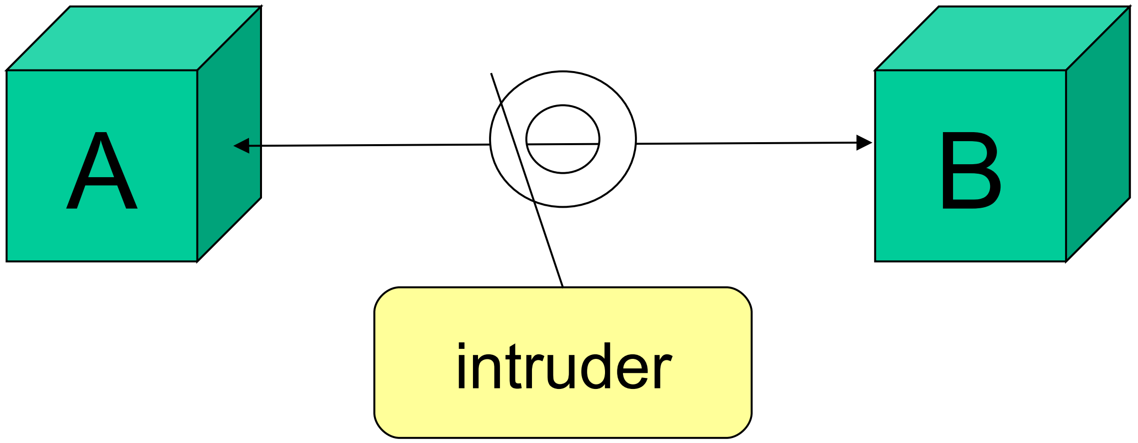
- Cryptography is the science and study of secret writing.
- Cryptanalysis is the science and study of methods of breaking ciphers.
- Cryptology: cryptography and cryptanalysis.
- Today [HAC]: Cryptography is the study of mathematical techniques related to aspects of information security, such as confidentiality, data integrity, entity authentication, and data origin authentication.

Origins of Cryptography

The enemy is an outsider listening to traffic

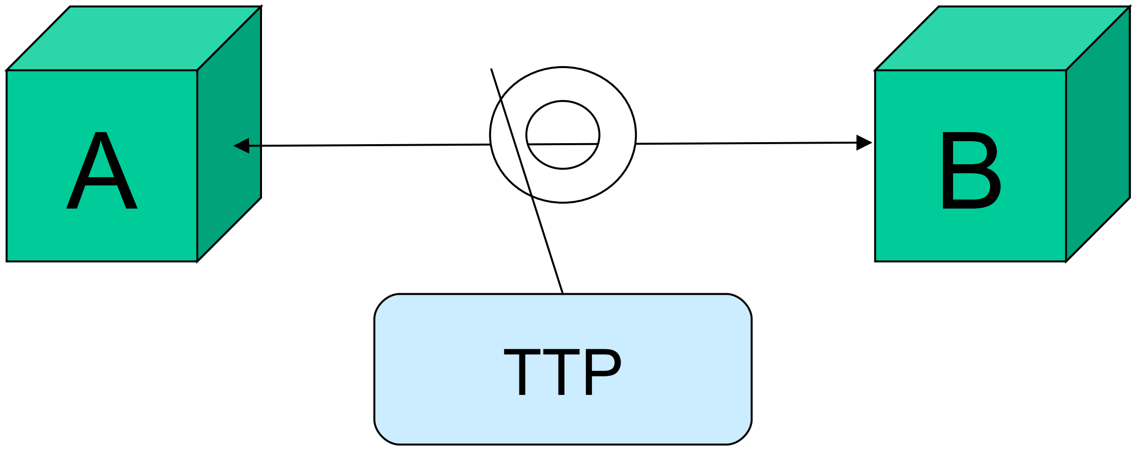
Two secure end systems communicate over an insecure channel

Old Paradigm



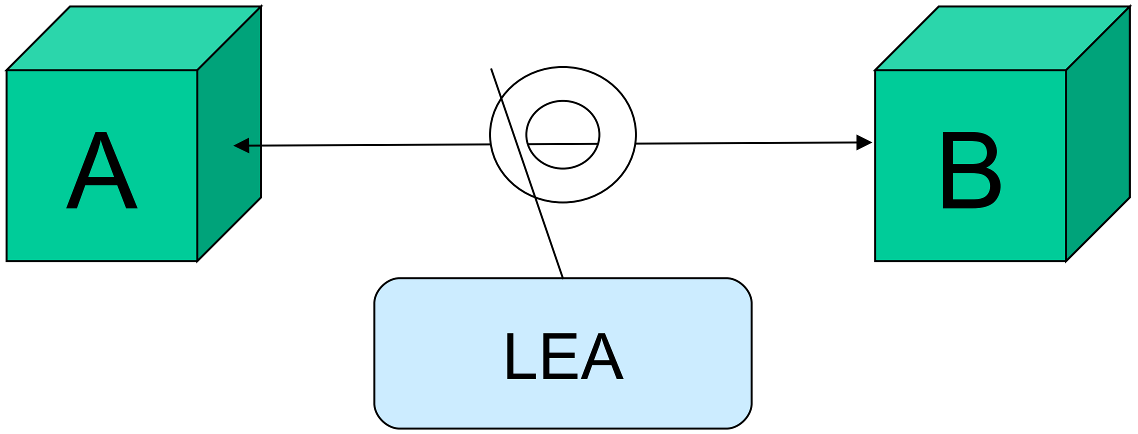
- *A* and *B* communicate over an insecure channel.
- *A* and *B* trust each other.
- Intruder can read, delete, and insert messages.
- With cryptography, *A* and *B* construct a secure logical channel over an insecure network.

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.

Cryptographic Keys

- Cryptographic algorithms use keys to protect data.
- Kerckhoffs' principle: do not rely on the secrecy of algorithms; the key should be the only secret that needs protection.
  - De facto standardisation and open evaluation of public algorithms is today the norm.
- Key management issues:
  - Where are keys generated?
  - How are keys generated?
  - Where are keys stored?
  - How do they get there?
  - Where are the keys actually used?
  - How are keys revoked and replaced?

Shifting the Goal Post

- Cryptographic keys are sensitive data stored in a computer system; access control mechanisms in the computer system have to protect these keys.
- Lesson: cryptography is rarely ever the solution to a security problem; cryptography is a translation mechanism, usually converting a communications security problem into a key management problem and ultimately into a computer security problem.

Crypto in Computer Security

- Vault for locking away secrets: unlocked with a key when putting data in or taking data out; implemented by symmetric encryption mechanisms.
- Transparent vault (cf. public lottery draws): everyone sees what is in the vault, a private key is need to fill it; a public key is the unique serial number of the vault.
- Private letter box: anybody can drop documents, only the owner can open it with a private key; a public key is the serial number of the letter box; like the feature above implemented using public key cryptography.
- When a document leaves your control, save a fingerprint so that you could detect any eventual later changes; can be implemented with hash functions.

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 $h(x) = y$.

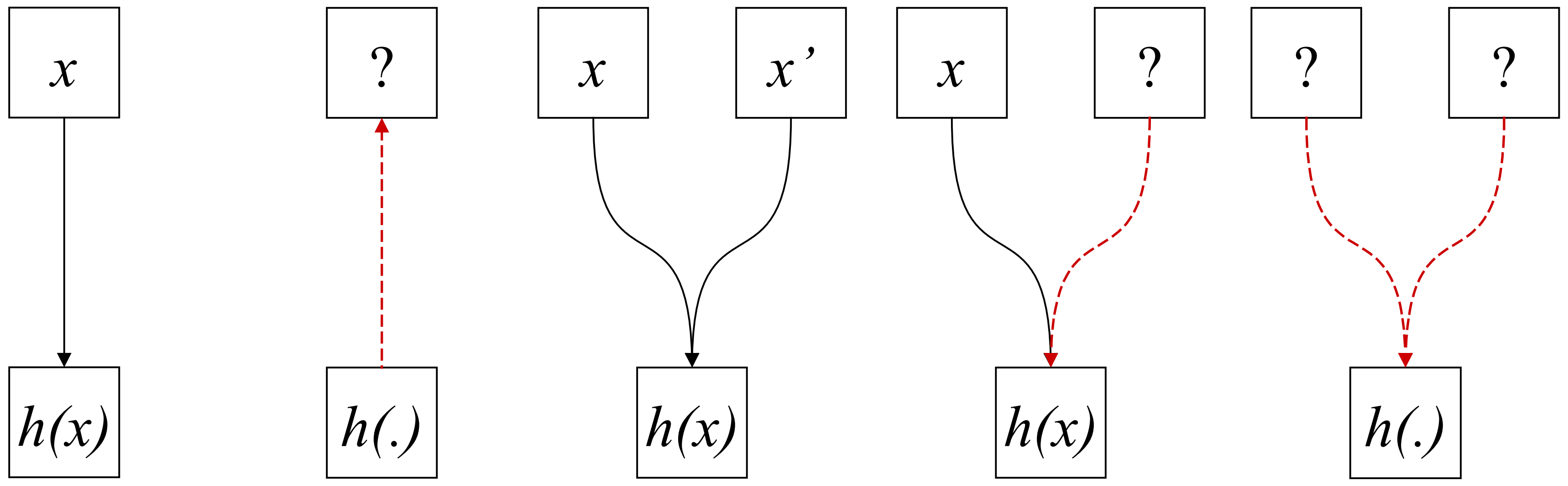
Collisions

- 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

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.

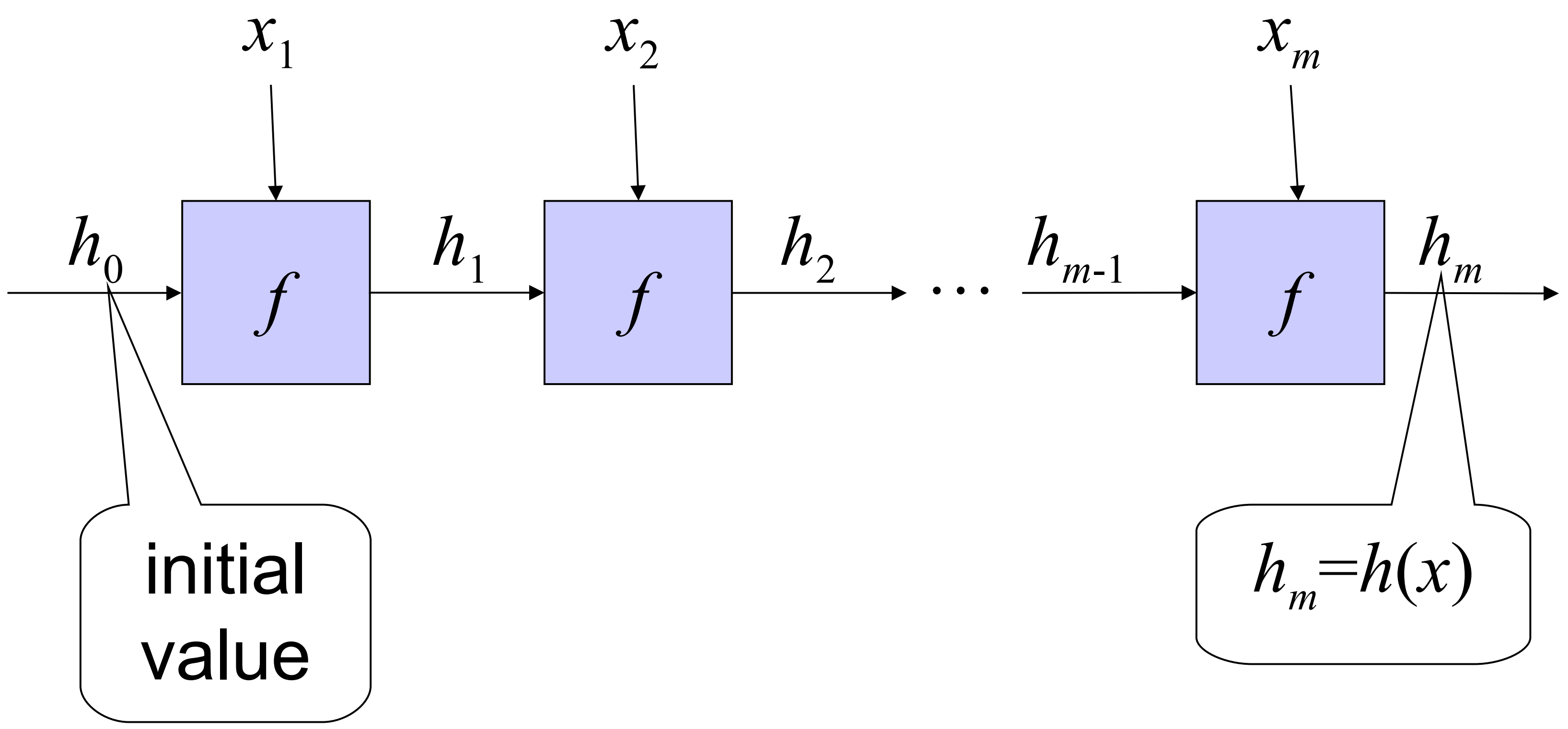
Checksums

- 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.

Construction

- Pattern for the design of fast hash functions:
- 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 $x_1,\dots, x_m$ of the given block size; last block has to be padded.
- Repeatedly apply the compression function: with a (fixed) initial value h_0, compute $h_i = f(x_i||h_{i-1})$ for $i=1, \dots, m$, 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, so broken and no longer recommended.
- Secure Hash Algorithm (SHA-1): designed to operate with the US Digital Signature Standard (DSA); 160-bit hash value; collision attacks reported.
- RIPEMD-160: hash function frequently used by European cryptographic service providers.
- SHA-256: when longer hash values are advisable.

Message Authentication Codes

- In communications, we cannot 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 verify a message, receiver has to share the secret key used to compute the MAC with the sender.
- 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.

HMAC (simplified)

- A MAC algorithm can be derived from a MDC algorithm h using the HMAC construction:
- For a given key k and message x, compute

  $$\mathrm{HMAC}(x) = h(k||p_1||h(k||p_2||x))$$

  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.
- Details of HMAC specified in RFC 2104.

Digital signatures

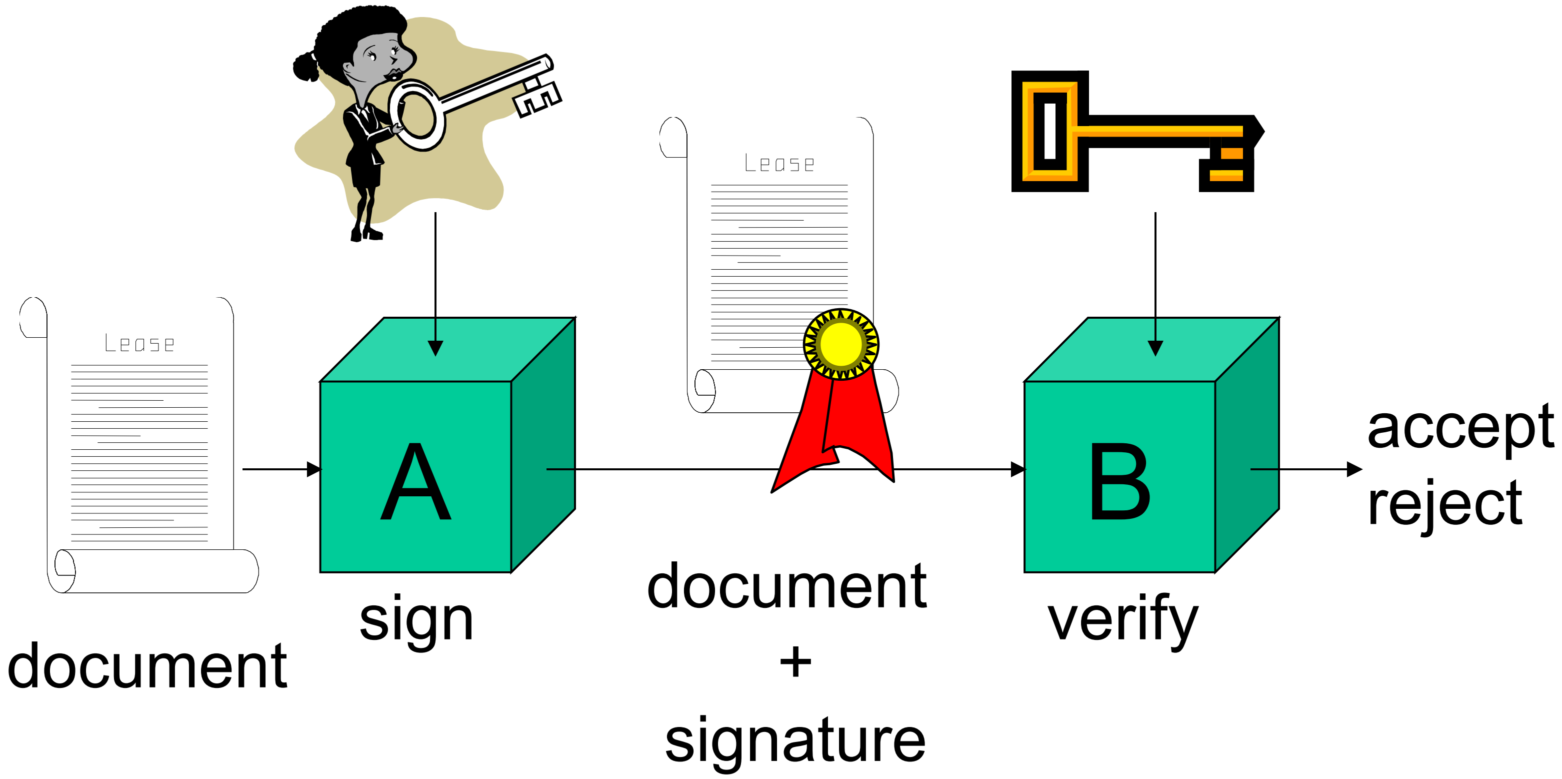
Digital Signature Mechanisms

- 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

- *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.
- 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.

Digital Signatures



RSA Signatures

- 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 $gcd(d,p\text{-}1) = 1$ and $gcd(d,q\text{-}1) = 1$.
  - Public verification key: $n = p\cdot q$ and an integer e with $e\cdot d = 1 \bmod \mathrm{lcm}(p\text{-}1,q\text{-}1)$.

RSA Signatures

- Signing: signer *A* hashes the document m so that $0 < h(m) < n$ and computes signature $s = h(m)^d \bmod n$.
- Verification: verifier uses a verification key (n,e) and checks $s^e \overset{?}{=} h(m) \bmod n$.
- For a correct signature, this equation holds because $s^e = h(m)^{d\cdot e} = h(m) \bmod n$.
- Hash function adds format check on message.
- Otherwise, existential forgeries are possible:
  - Pick signature s, construct 'message' $m = s^e \bmod n$.
  - m is random bit string; can be detected by format check on m.

Performance Gains

- Signature verification is particularly quick for specific ‘small’ public verification keys, e.g. $e = 2^{16}+1$.
  - Performance measurements for RSA often show much smaller values for verification than for signing; in such cases a ‘small’ public key had been used.
- Signatures with message recovery: RSA has modes where short documents can be recovered from the signature and do not have to be transmitted separately.
  - Relevant e.g. for smart card applications.

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 $n = p \cdot q$ is “difficult”.
- When the public modulus $n = p \cdot q$ 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.

MACs & Digital Signatures

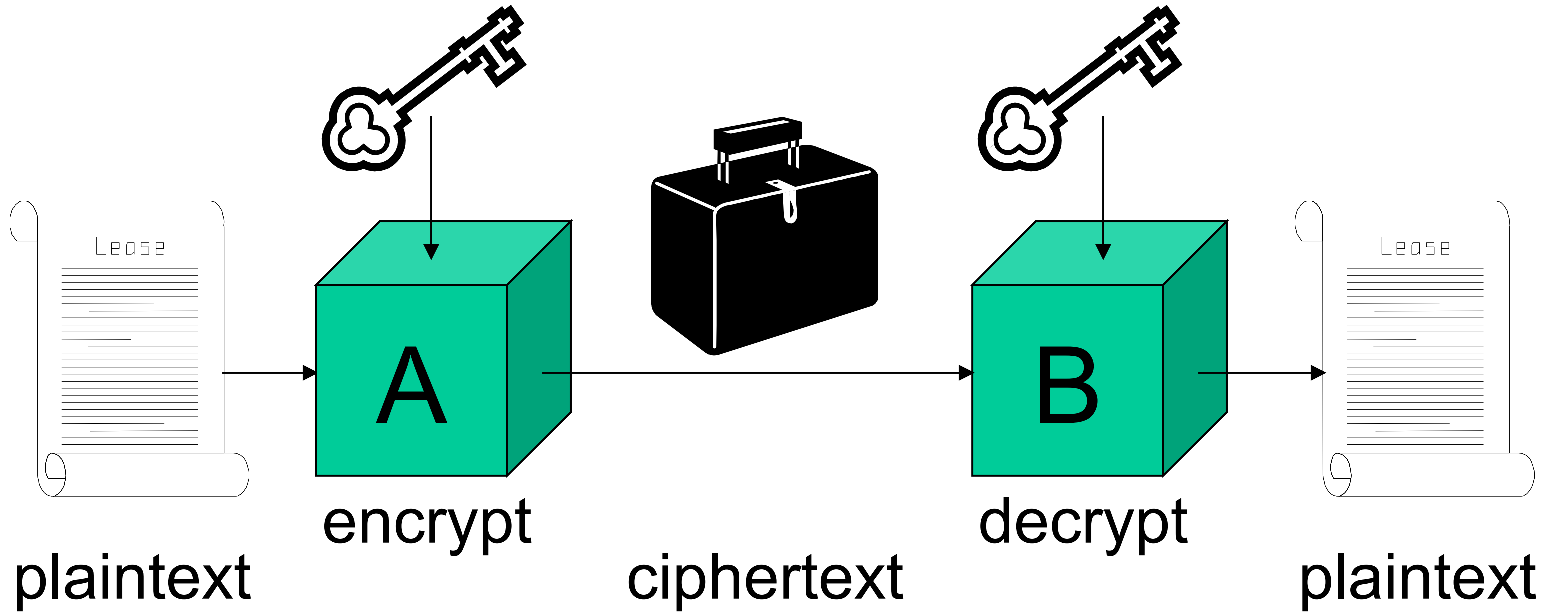
- MACs and digital signatures are authentication mechanisms.
- MAC: verifier needs the secret used to compute the MAC; MAC unsuitable as evidence with a third party.
  - Third party would need the secret.
  - Third party cannot distinguish between the parties knowing the secret.
- Digital signatures can be used as evidence with a third party.
- The term "non-repudiation" was coined to distinguish the features of authentication based on digital signatures from MAC-based authentication.

Encryption

Terminology

- Encryption: plaintext (clear text) x is converted into a ciphertext under the control of a key K.
    - We write $\mathrm{e}K(x)$.
- Decryption with key K computes the plaintext from the ciphertext y.
    - We write $\mathrm{d}K(y)$.
- 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.

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

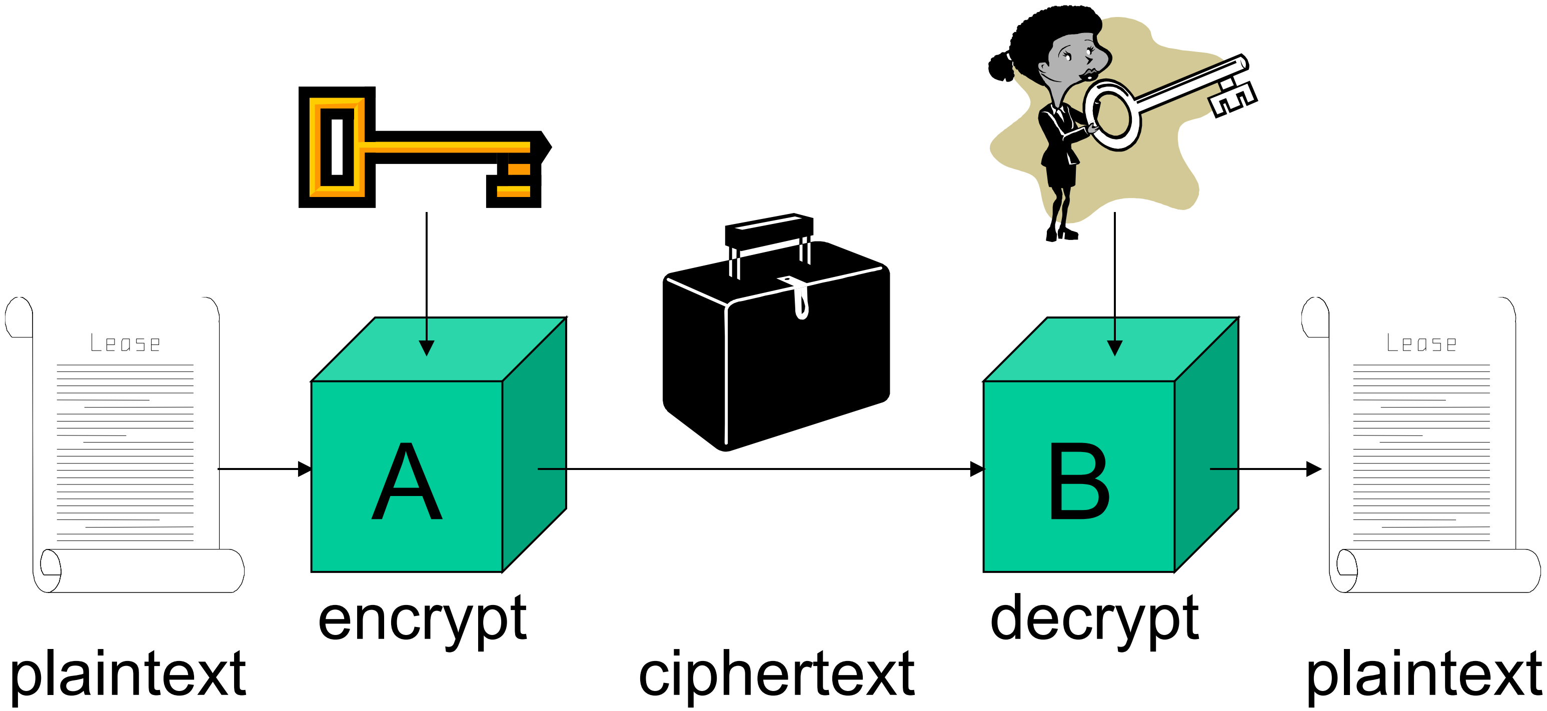
Advanced Encryption Standard

- Public competition to replace DES: 56-bit keys and 64-bit data blocks no longer adequate.
- Rijndael (pronounce as “Rhine-doll”) nominated as the new Advanced Encryption Standard (AES) in 2001 [FIPS-197].
- Designed by Vincent Rijmen & Joan Daemen.
- Versions for 128-bit, 192-bit, and 256-bit data and key blocks (all combinations of block length and key length are possible).
- Rijndael is not a Feistel cipher.

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

- 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.
- RSA encryption 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 $gcd(e,p\text{-}1) = 1$ and $gcd(e,q\text{-}1) = 1$.
  - Private decryption key: an integer d with $e[\quad d = 1 \bmod \mathrm{lcm}(p\text{-}1,q\text{-}1)$.

RSA Encryption

- Messages are broken into message blocks m_i so that $0 < m_i < n$.
- Encryption: sender *A* takes a message block m and computes the ciphertext $c = m^e \bmod n$.
- Decryption: receiver uses its decryption exponent d and computes $m = c^d \bmod n$.
- Note: $c^d = m^{e\cdot d} = m \bmod n$.
- 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.