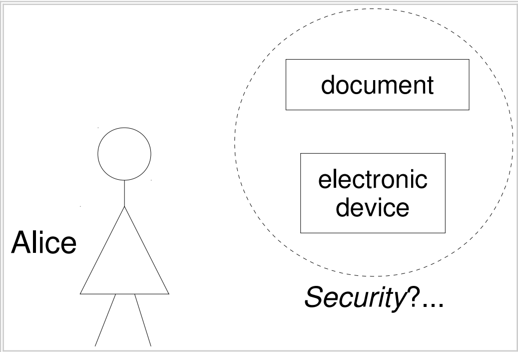
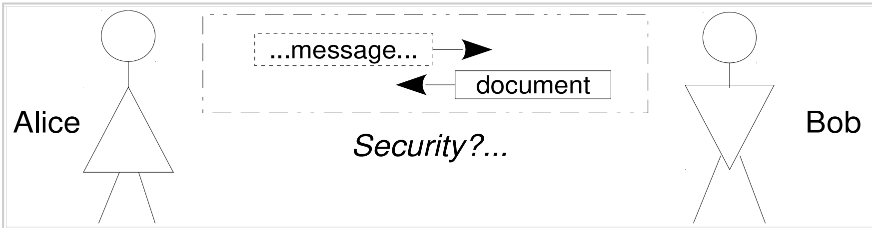
COMPUTER SECURITY

Cryptography: general protection techniques





Protection basics

Purpose

- provide **access control** to resources (e.g. users' information)
  - by building secure channels
    - for communication
    - for storage
  - with properties
    - main: confidentiality, integrity and authentication
    - secondary: anonymity, forward secrecy, etc.

Secure channel

- cryptographically-protected conversation line between two identified subjects
  - called, in some contexts, *security association*
- basic, expected properties:
  - Authentication – assuring that each subject is talking to the genuine other
  - Integrity – assuring that deletion, change or addition of data is detected
  - Confidentiality – assuring that data is understandable only by subjects

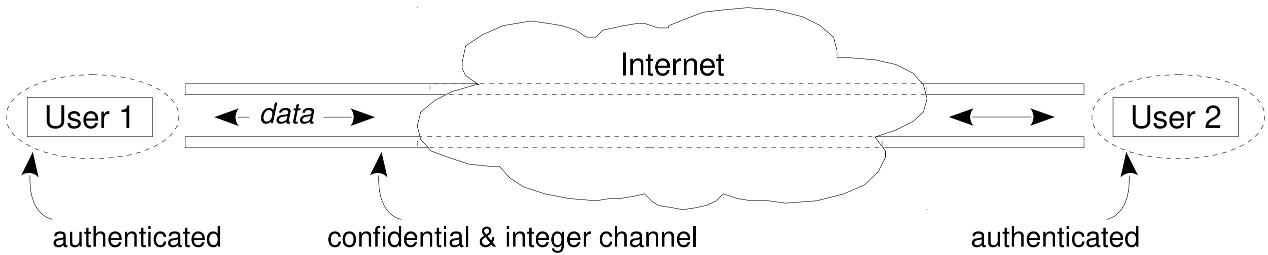


Fig. General secure (communication) channel.

Utilization of secure channel

- 1st: Authentication of one or both subjects and probable parameter negotiation
  - usually,
    - an asymmetrical cipher is used
    - a "session key"[1] is created
- 2nd: Utilization proper
  - maybe also with protection for
    - confidentiality
    - integrity
  - usually,
    - a symmetrical cipher is used (with above session key)

1 more on this, elsewhere

Protection Properties

Authentication

- assuring the identity of the entities involved
  - binding identifiers to subjects
- sometimes: certifying a location
  - e.g. machine's[1] in the Net, machine's or user's geographical position
- authentication system's operation: two phases
  - setup: generation and storage of subjects' authentication data in system
    - repeated whenever user changes own authentication data
  - usage: normal procedure for authentication of subjects
- authentication operation: two steps
  - presentation (of subject) [sometimes: *identification*[2]]
  - validation (proof of authenticity) [sometimes: *authentication*]

1 origin of a communication...
2 Note: this occasional use of “identification” is unfortunate. In reality, identification is the process of binding an identifier to an individual, as yet unknown (i.e. for whom no label, or name, was yet presented).

Remote authentication

- user's physical intervention is not possible (or required)
  - presentation by non-physical identifier
  - validation by *proof of knowledge*, typically of challenge-response type
- based on the use of pre-distributed keys
- generally, use *nonces*

Nonce:

- piece of data that is both:
  - fresh
  - not guessable (random)
- random number generated when about to be used
- used for binding two messages in a challenge-response sequence

...Protection Properties: Authentication...

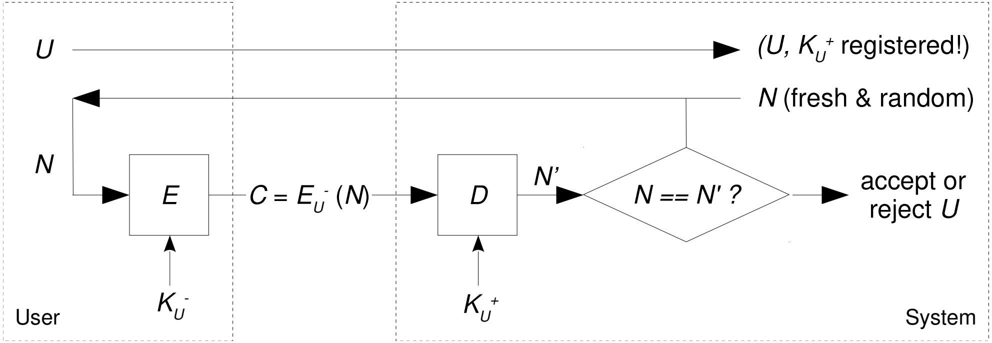
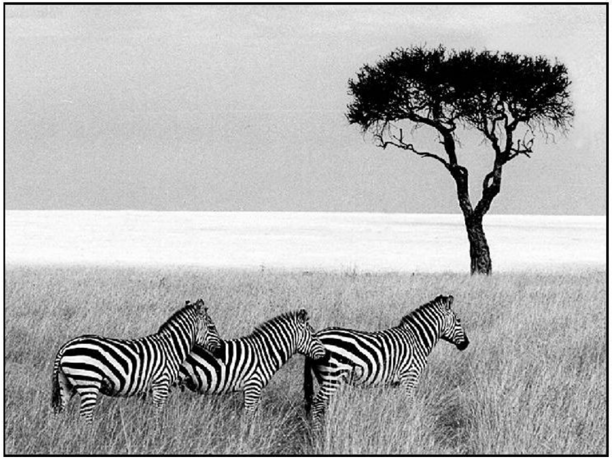


Fig. Authentication with asymmetric cryptography.

Confidentiality

- assurance of limited disclosure of information
  - implies Authentication of the entities involved!



Solutions

- hide the sensitive documents
  - physically saving them
  - cunningly disguising them
    - steganography! [FIG[1]]
- encipher documents
  - parties need appropriate keys

1 Presumably, the original of this picture (coloured, 1024×768 pixel), contains in compressed form the complete unabridged text of five Shakespeare's plays, totaling more that 700kB of text. (Tanenbaum, Modern Operating Systems)

Hiding of documents

- not covered here (see steganography examples in the literature)

Encipherment of documents

- symmetrical technique
- asymmetrical technique

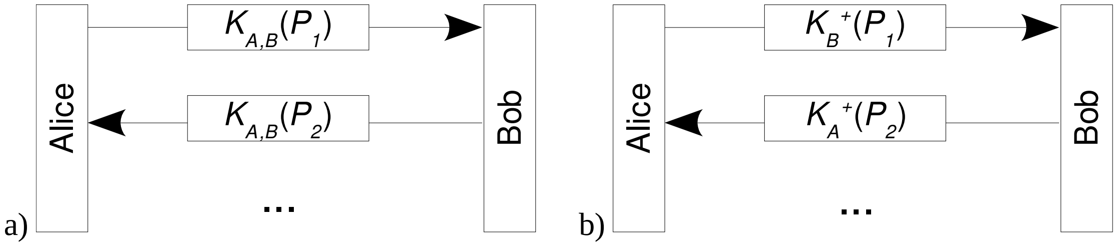


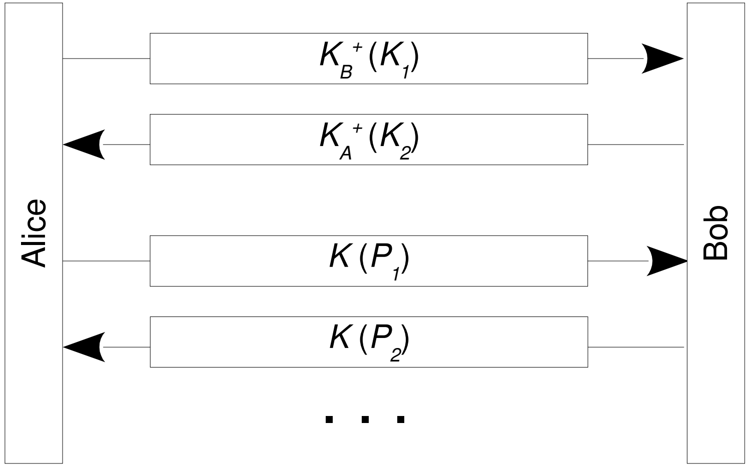
Fig C. Base encipherment techniques: a) shared key; b) public key.

...Protection Properties: Confidentiality...

Practical problems:

- durable symmetric keys are difficult to manage
- asymmetrical operations are very inefficient

- So, usual solution is a mix:[1]
  1. exchange symmetric key by public-key means
  2. encipher documents with exchanged shared (ephemeral) key



$K = public\_function\ (K_1, K_2)$

Fig. Usual solution for a confidentiality-protected communication channel.

1 Conceptually, techniques for following the steps are sometimes called: 1. key encapsulation mechanism (KEM) ; 2. data encapsulation mechanism (DEM).

Integrity

- assurance that a change in "document"[1] is detected[2]
  - implies Authentication of the entities involved!

Solutions

- encipher the document[3]
  - with symmetric or asymmetric algorithms
- use integrity code
  - with shared key
- digitally sign the document
  - directly, with private key
  - through its digest, with private key

1 file, message,...
2 if detected, abusive change cannot (in general) be reversed (corrected)
3 In reality, this is not a (good) solution!

Simple "solution" for integrity problem: encipher everything!

- exchange ciphered information
  - detection of alteration of message[1]
  - confidentiality also granted[2]

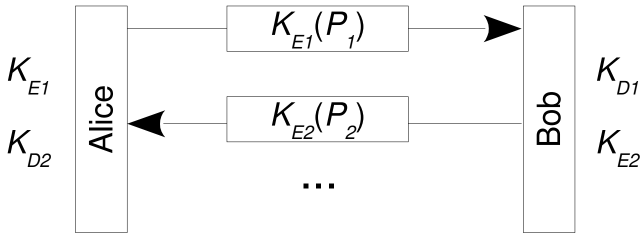


Fig. A *not* real solution for the integrity protection of a communication channel.

Problems

- symmetric cipher: no origin authenticity (repudiation is possible)!
- asymmetric cipher: low efficiency!
- in any case, alterations can go unnoticed:
  - in applications with general binary data (numbers, pictures...)
  - with some confidentiality algorithms that do not guarantee integrity[3]

1 e.g. because intelligibility is affected
2 but not relevant here
3 e.g. One-time pad

Better solution: use Message Integrity Codes, MIC[1]

- parties agree on a (shared) key
- sender builds an *hash* of "message *plus* key":[2] that is the MIC!
  - e.g. MIC = $h\ (m \,\|\, K)$, where $\|$ means concatenation
- sender transmits both message and MIC
- receiver checks message's integrity, by repeating hash with knowledge of the key

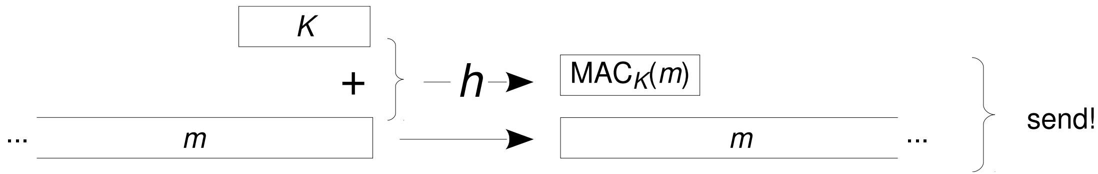


Fig. General construction principle and usage of Message Integrity/Autentication Codes.

1 The *Message Integrity Check* term (RFC 1421), is currently not much used and said deprecated in RFC 4949; the designation in fashion is *Message Authentication Code*, MAC. Some authors make a slight distinction between the two (e.g. Menezes et al. in *Handbook of Applied Cryptography*); I will not and prefer MIC, as I find it more clear.
(A related term, *authenticator*, was probably first used in 1983 (Davies and Clayden) in the description of a *Message Authenticator Algorithm*.)

2 *keyed hash* technique

...Protection Properties: Integrity with message integrity codes...

Problems

- uses a shared key
  - parties must exchange it, somehow
  - there is no prevention for:
    - message alteration or forging by the recipient
    - message repudiation by the sender!

Exercises:

- Usually, hashing is an iterated calculation of message blocks. That might enable the emergence of vulnerabilities:
  - what vulnerability would readily turn up if in the *keyed hash* technique MIC/MAC was instead defined as $h\ (K \| m)$?
  - however, even with format $h\ (m \| K)$, there is a problem if one can find a hash collision: for m != m' , $h\ (m) = h\ (m')$. Verify it.

A secure MIC: the HMAC

- HMAC, *Hashed Message Authentication Code*, IETF RFC 2104
  - HMAC (H, K, m) = H {($K \oplus$opad) || H [($K \oplus$ipad) || m)]}

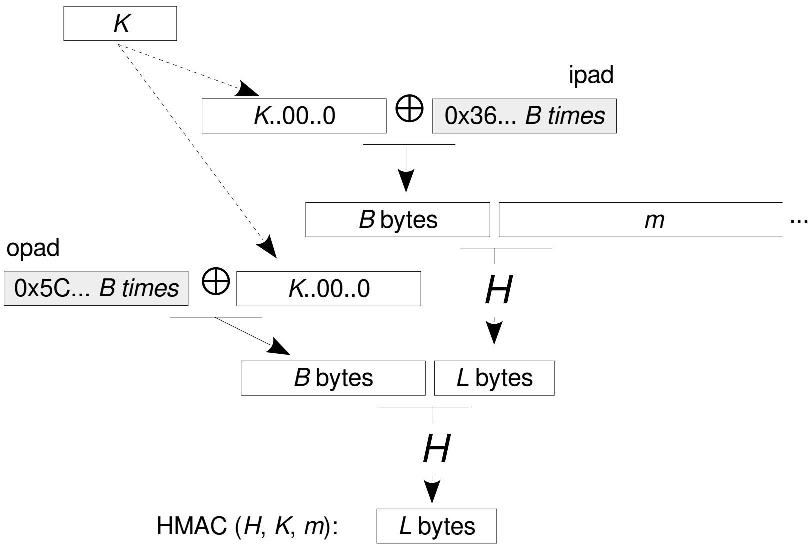


Fig. HMAC operation: H = hash function, L = hash size, B = block size of hash function iteration.

For H = SHA-1: L = 20 bytes, B = 64 bytes.

E.g.: HMAC (SHA1, "key", "message") = 2088df74d5f2146b48146caf4965377e9d0be3a4

Great solution: use digital signatures

- allow:
  - checking of a document for alteration
  - associating a document to its author
- and so:
  - only author can change the original document
  - readers are assured of the identity of author
  - author is not able to deny authorship of document (repudiate it)

Techniques

- public key[1]
- message digest (with public key!)

1 In reality, a digital signature is made with a *private* key!

Digital signatures: (plain) public key technique

- encipherment with sender's private key
- decipherment with sender's public key

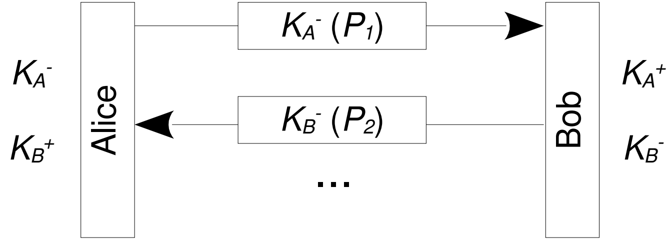


Fig. Integrity protection with digital signatures: plain public-key technique.

Problems (plain technique)

- "major":
  - asymmetric cipher: low efficiency!
- "minor":
  - sender's private key must be kept secret
  - sender's public key must be known in advance
  - longevity of protection of sent document implies safe keeping of key pair

Digital signatures: message digest (with public key) technique

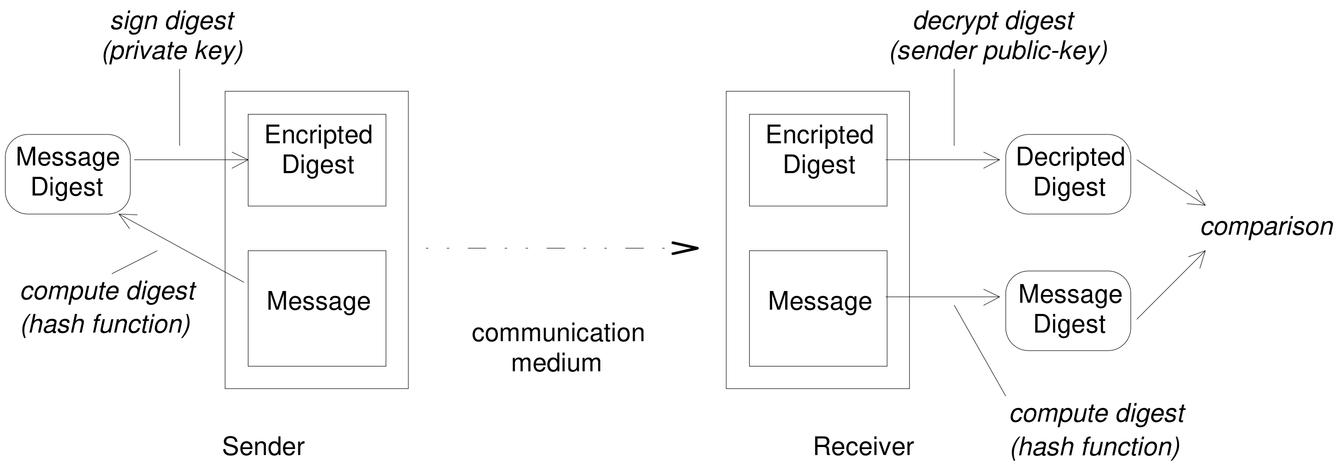


Fig. Integrity protection with digital signatures: message digest technique. (*in* Tanenbaum, ...)

...Protection Properties: Integrity with digital signatures...

Problems (message digest technique)

- "major":
  - greater complexity[1]
  - hash function should be collision-free
- "minor":
  - same as (simple) public key's technique

Exercises:

- Verify that this technique[2] prevents attacks of the types:
  - "existential forgery", whereas someone can produce a signature for a message (whose content might be of no relevance) and say it is of someone else.[3]
  - "specific forgery", whereas someone can produce a signature for a message related to other known signed messages.[4]

1 but without significant efficiency penalty as: hashing is very fast; public-key operations are on few bytes (e.g. 32 B with SHA-256)
2 contrary to the (plain) public-key signature
3 For example, if P is signed with $K_E^-(P) = S$ and both (P, S) were sent to a receiver, an attacker could forge E's signature for another message by choosing a signature S_M, doing $K_E^+(S_M)$ to get P_M and sending both (P_M, S_M) to the receiver!
4 E.g. for RSA: if $P1$, $K_E^-(P1)$ and $P2$, $K_E^-(P2)$ are known, a forgery of $P1*P2$'s signature could be performed, as the property $K_E^-(P1) * K_E^-(P2) = K_E^-(P1*P2)$ is verified in RSA.

Attacks on digital signatures[1]

Goal:

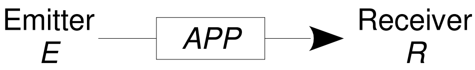
- forge the signature of a new message or, preferably, grasp the signing key

Possible approaches:

- normal
  - only some few messages and their signatures are available
- known original text ("passively" obtained)
  - for a variety of known messages, their signatures are available
- planned original text ("actively" prepared)
  - specific chosen messages are made to be signed
- vulnerable fingerprinting function (digest method)
  - find hashing collisions to help with previous approaches

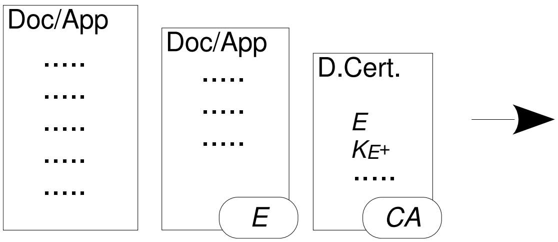
1 Extension to the section *Breaking cryptographic systems*, presented in a previous chapter.

Example: Secure distribution of documents or software



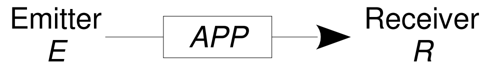
Part I: Emission

- Emitter E of application/document APP
  - digitally signs APP
    - usually, digest technique...
    - generates $[APP]_E$ [1]
  - appends to $[APP]_E$ a digital certificate $[DC(E)]_{CA}$
    - certificate has K_E^+
    - is signed by CA[2]
  - sends everything to Receiver
    - $APP + [APP]_E + [DC(E)]_{CA}$



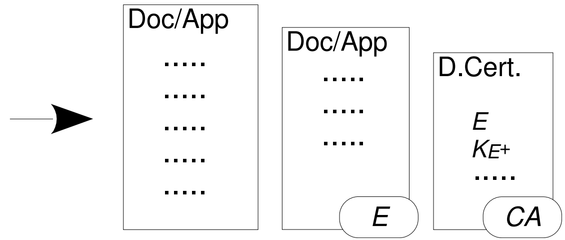
1 Notation of digital signature: $[DOC]_E$ <==> K_E^- (DOC) or $[DOC]_E$ <==> K_E^- (h(DOC))
2 also trusted by Receiver!

Example (cont.): Secure distribution of documents or software



Part II: Reception

- Receiver *R* of application/document
  - gets K_E^+ of Emitter[1]
    - by processing the digital certificate $[DC(E)]_{CA}$
      - must already know, or somehow get, K_{CA}^+
      - checks the integrity of $[DC(E)]_{CA}$
  - checks the integrity of $[APP]_E$
  - uses *APP* with confidence!



1 if it was not available previously

Integrity + Confidentiality

- confidentiality protection does not guarantee integrity protection
- so, some type of integrity protection must be added
  - basic example: combine secrecy with digital signatures
  - in general: use *authenticated encipherment* modes (to be seen)

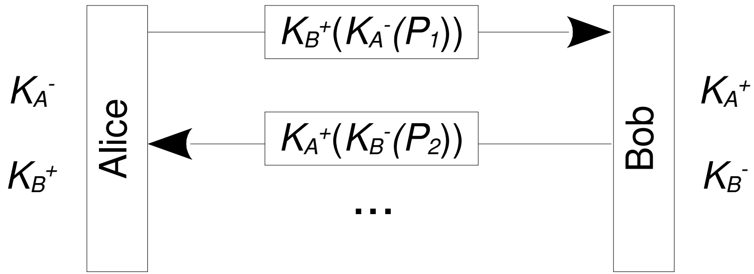


Fig. Confidentiality and integrity protection of a communication channel: basic solution (not considering efficiency).

Pointers...

- **Steganography: Hiding Data Within Data**, 2001 – Gary Kessler
  - www.garykessler.net/library/steganography.html
- The "**HMAC RFC**", 1997 – H. Krawczyk, M. Bellare, R. Canetti
  - tools.ietf.org/html/rfc2104
- The "**Handbook of Applied Cryptography**"- 1 ed, 1996 – A. Menezes, P. van Oorschot, S. Vanstone, CRC Press, *Chap. 11 - Digital Signatures*, p. 425
  - freecomputerbooks.com/handbook-of-applied-cryptography.html
- "**Authenticated encryption**", -2024 – Wikipedia
  - en.wikipedia.org/wiki/Authenticated_encryption