COMP3260/6360 Data Security

Lecture 9

Prof Ljiljana Brankovic

Lecture Overview

1. Key Management
   a) Distribution of public keys
      i. Public Announcement
      ii. Publicly Available Directory
      iii. Public-key Authority
      iv. Public-Key Certificates
   b) Public-Key Distribution of Secret Keys

2. Message Authentication
   a) Encryption
   b) Massage Authentication Code
   c) Hash functions

Key Management

Chapter 14 from text: Key Management and Distribution

Chapter 11 from text: Cryptographic Hash Functions

Chapter 12 from text: Message Authentication Codes

Note that in-text references and quotes are omitted for clarity of the slides. When you write as essay or a report it is very important that you use both in-text references and quotes where appropriate.

Key Management

We discuss:

- Distribution of public keys
- The use of public-key encryption to distribute secret keys

Four main categories of techniques for distribution of public keys:

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

We use the following notation from the textbook:

KU_a - public key of user A

KR_a - private key of user A

E_{KUa} - encryption under A's public key

E_{KRa} - encryption under A's private key

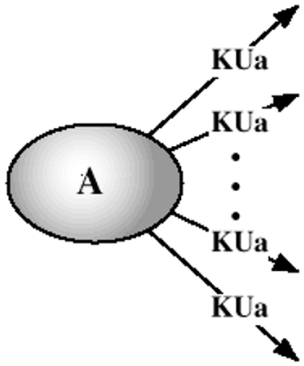
D_{KUa} - decryption under A's public key

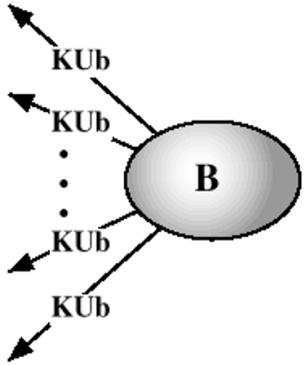
D_{KRa} - decryption under A's private key

Public Announcement

- Unlike in conventional cryptosystems, in public-key cryptosystems public keys are public knowledge, and thus there is no need for confidentiality in their distribution. However, there is still a need for authenticity/integrity. The private keys in public-key cryptosystems need not be distributed at all.
- An obvious way to distribute public keys is public announcement, where participants simply send their public keys to any other participants, or broadcast the keys to the community.
- For example, users of PGP (Pretty Good Privacy) may include their public keys in email messages, publish them on their Web pages, etc.

The main drawback of this approach is that authenticity is not provided: anyone can forge such a public announcement. In other words, user X can pretend to be user A and send their public key to another participant or broadcast it to the community. By the time A discovers the forgery, lots of damage could be already done.





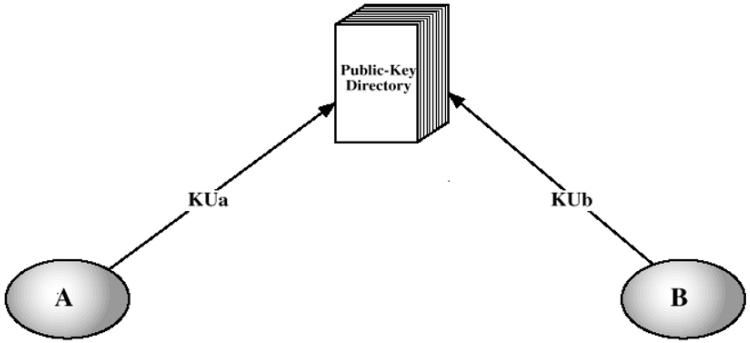
Publicly Available Directory

Under this scheme the public keys are kept in a public directory maintained by a trusted authority.

- The authority maintains a directory with a {name, public key} pair for each participant.
- Each participant registers their public key with the authority - the registration has to be authenticated in some way (e.g., in person).
- Each participant may replace their key at any time.
- Periodically, the authority publishes the directory or updates to the directory; alternatively, participants can access the directory electronically – secure communication between the authority and the participants is essential!

The main drawbacks of this approach are:

- an adversary can tamper with the records, or can compromise the private key of the authority;
- a need for constant updates of the directory.



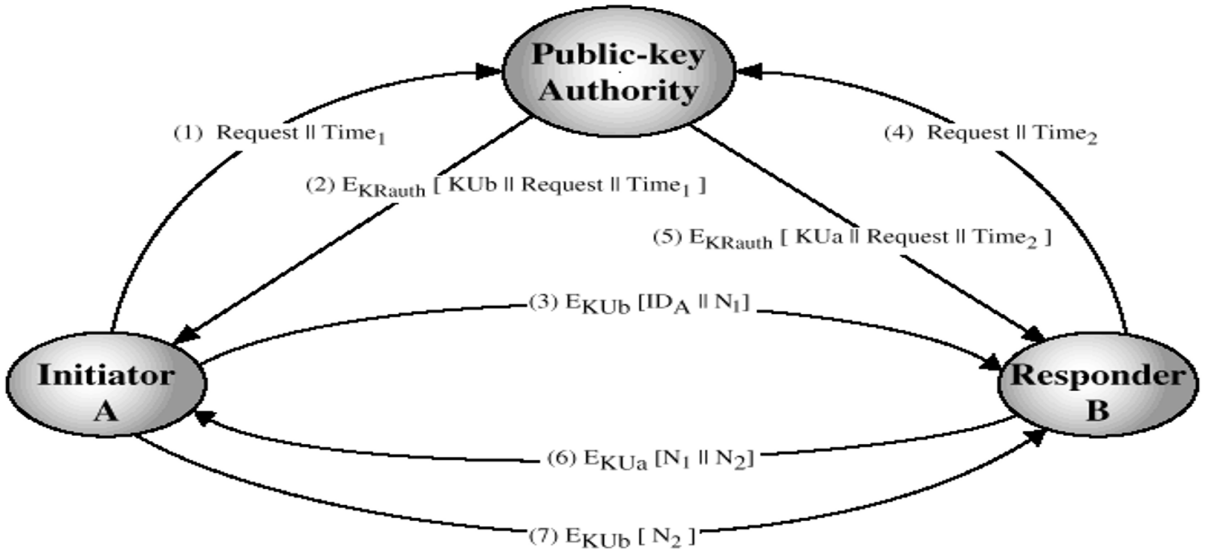
Each participant reliably knows the public key of the authority (and obviously, only the authority knows the corresponding private key).

Public-Key Authority

1. A sends to the authority a request for B's public key, together with a timestamp.
2. The authority responds with B's public key, the original request and the original timestamp; the whole message is encrypted using authority's private key KR_{auth}
3. A stores B's public key and sends to B their identity ID_A and a nonce N_1, used to uniquely identify the transaction; the whole message is encrypted using B's public key.
4.-5. B retrieves A's public key from the authority in the same way that A retrieved B's key.
6. B sends to A nonce N_1 as well as a nonce N_2 generated by B; the whole message is encrypted with A's public key; presence of N_1 in the message assures A that the correspondent is B.
7. A returns N_2 encrypted with B's public key, to assure B that the correspondent is A.

The main drawback of public-key authority system is that the authority is a bottleneck in the system as every user has to apply to the authority for every public key it wishes to obtain.

Additionally, the authority is vulnerable to tampering.



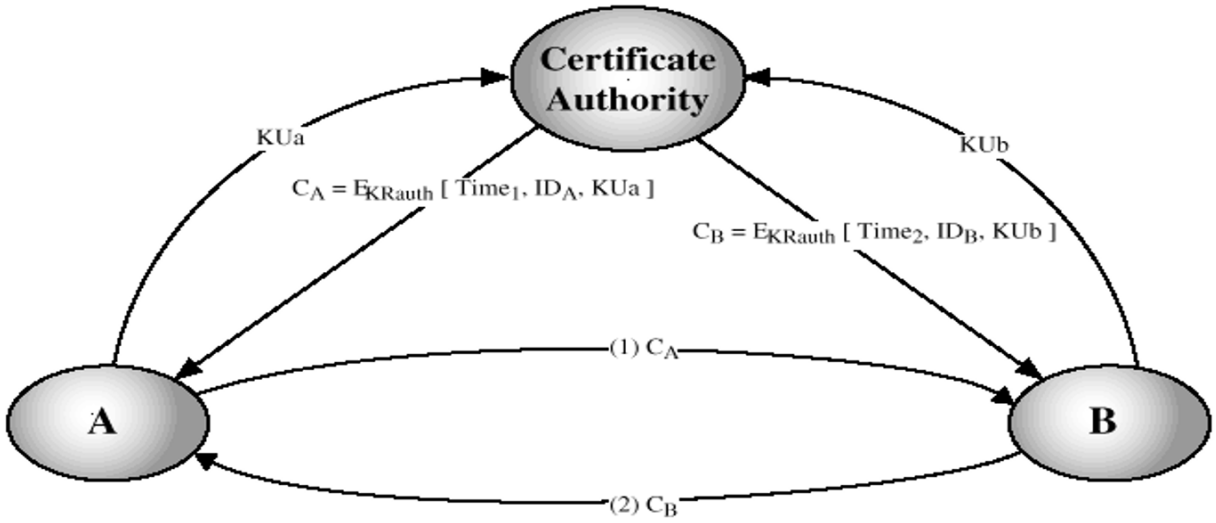
Public-Key Certificates

The main idea in this scheme is that participants can exchange keys without contacting the authority, while the keys are as reliable as if they were obtained from the authority.

Each user has a certificate that is created by the authority and that contains the user's public key together with some additional information.

The following requirements have to be satisfied:

- Any participant can read a certificate to determine the name and public key of the certificate's owner.
- Any participant can verify that the certificate was created by the authority.
- Only the certificate authority can create and update certificates.
- Any participant can verify the currency of the certificate.

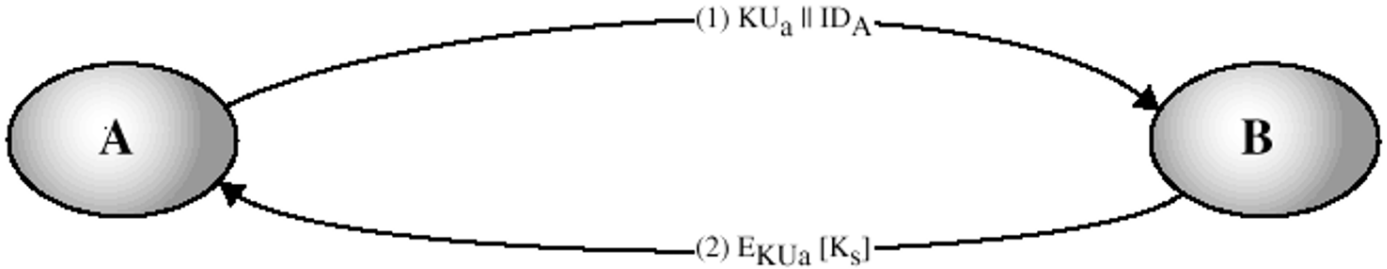


Public-Key Distribution of Secret Keys

Public-key encryption is slower that the conventional encryption, and for that reason it is typically not used for communication; rather, it is used for distribution of secret keys to be used in conventional encryption.

Simple Secret Key Distribution

- A generates a public\private key pair $\{KUa, KRa\}$ and sends a message to B containing KUa and IDa (identifier of A).
- B generates a secret key Ks, and send it to A encrypted with A's public key.
- A computes $D_{KRa}[E_{KUa}[Ks]]$ to recover the secret key Ks. Note that only A can do this, thus only A and B will know Ks.
- A discards KUa and KRa and B discards KUa.

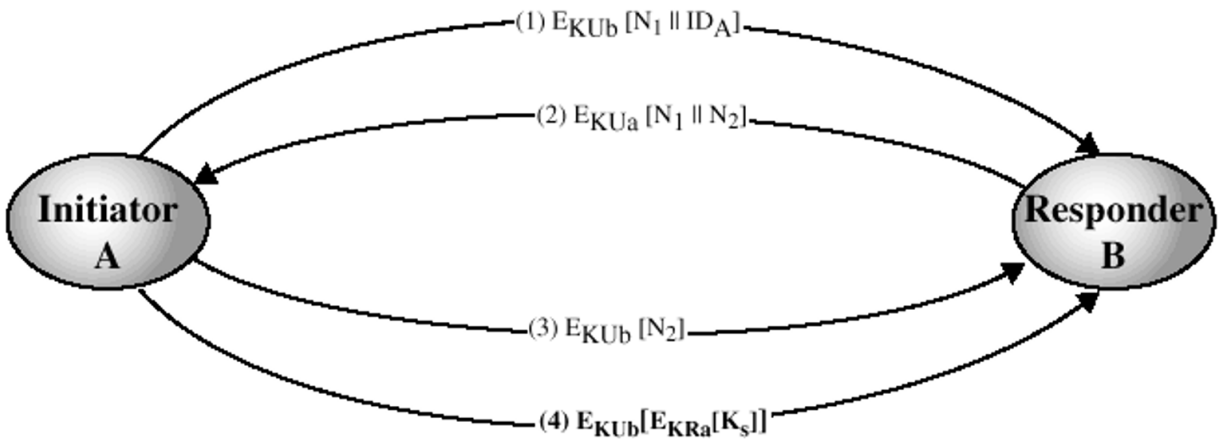


Simple Secret Key Distribution

The main drawback of this scheme is that it is vulnerable to man-in-the-middle attack.

- A generates a public\private key pair $\{KUa, KRa\}$ and sends a message to B containing KUa and IDa (identifier of A).
- E intercepts the message, creates their own public\private key pair $\{KUe, KRe\}$ and transmits KUe and IDa to B.
- B generates a secret key Ks and transmits $E_{KUe}[Ks]$
- E intercepts the message and learns Ks by computing $D_{KRe}[E_{KUe}[Ks]]$
- E transmits $E_{KUa}[Ks]$ to A.
- From now on E simply eavesdrops, while A and B are not aware of the problem.

Secret Key Distribution with both **confidentiality** and **authenticity**:



A Hybrid Scheme

In this scheme a Key Distribution Center (KDC) shares a master key with each user and distributes secret session keys encrypted with the master key. A public-key scheme is used to distribute the master keys.

Advantages:

- There are many applications in which the session keys change frequently; distribution of session keys with public-key encryption could degrade the overall performance; in this scheme, it is used only occasionally to update the master keys.
- The hybrid scheme is easily added on top of existing KDC.

Message Authentication

What if we do not need to keep a message secret and just want to verify the integrity of the message?

Message authentication is concerned with:

- protecting the integrity of a message
- validating identity of originator

Digital signature is concerned with non-repudiation of origin (dispute resolution).

Three different functions used:

- message encryption
- message authentication code (MAC)
- hash function

Types of attacks:

- Disclosure
- Traffic analysis
- **Masquerade**
- **Content modification**
- **Sequence modification**
- **Timing modification**
- Repudiation (source and destination)

The first two attacks are considered to be the matter of confidentiality, the next four the matter of authentication and the last one non-repudiation (digital signatures).

Authentication Functions

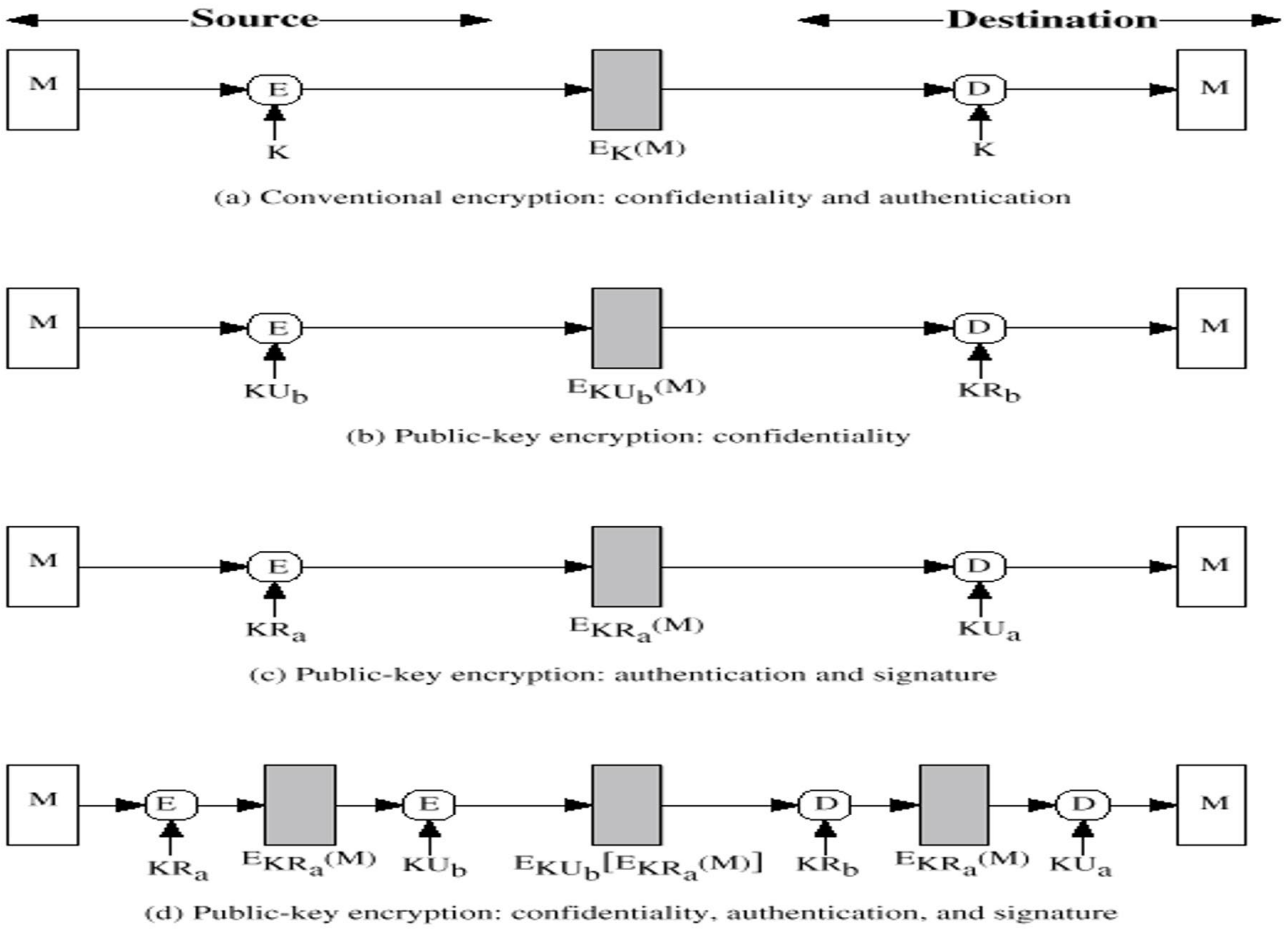
Any authentication system has two levels:

- lower level - authentication function that produces an authenticator (a value used to authenticate the message)
- higher level - authentication protocol

Authentication functions fall into the following classes:

- **message encryption** – the authenticator is the ciphertext of the entire message
- **message authentication code (MAC)** - the authenticator is a fixed-length value produced from the message and a key by a public function
- **hash function** - the authenticator is a fixed-length hash value produced from a message of any length by a public function

Message Encryption



If public-key encryption is used, encryption provides no confidence of sender since anyone potentially knows public-key.

However, if

- sender **signs** message using their private-key
- then encrypts with recipient's public key,

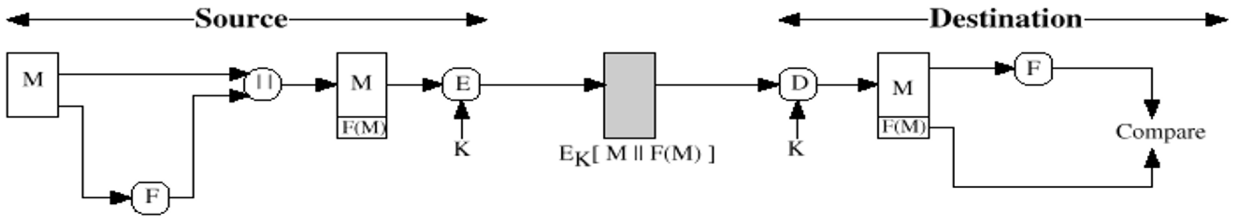
we have both confidentiality and authentication (and signature!), but at cost of two public-key uses on message

Message Encryption

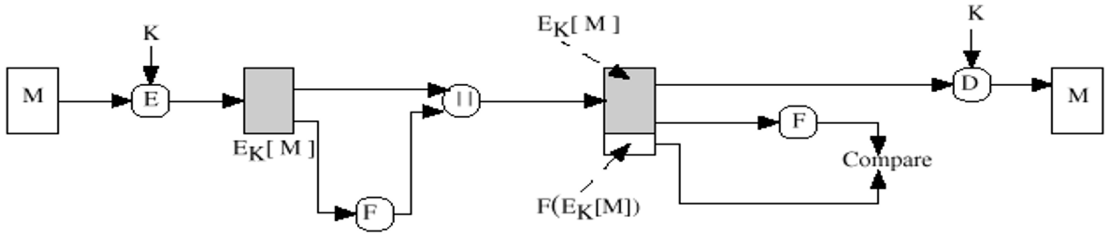
To be able to determine whether a message is a ciphertext of a legitimate plaintext, we require that only a small subset of all massages constitute a set of legitimate plaintexts. This requirement is satisfied more often than not, and certainly for an English text.

To be able to automatically determine whether a received message is a ciphertext of a legitimate plaintext, we append to the message an error-detecting code, or checksum, as illustrated on the next figure.

Error control can be internal or external, where external enables opponent to create messages with valid error-control codes and create confusion.



(a) Internal error control



(b) External error control

Message Authentication Code

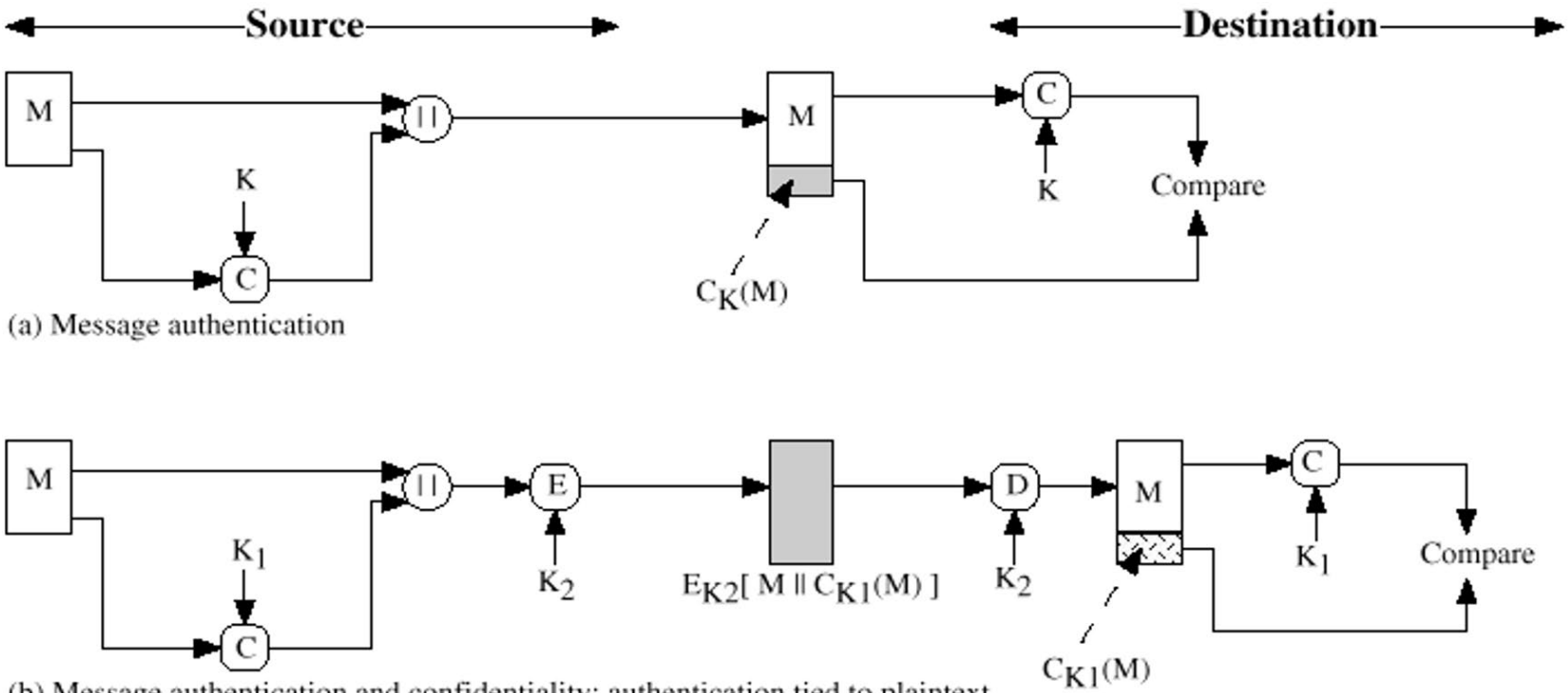
A secret key, known only to the sender and the receiver, is used to generate a small fixed-sized block of data (called cryptographic checksum or MAC), which is appended to the message. The receiver uses the same key to obtain the MAC and compare it to the received MAC.

The receiver is assured that:

- the message has not been altered
- the message comes from the sender
- if the message includes a sequence number, the receiver also knows that the ordering is correct.

We can use encryption to add confidentiality, as illustrated in the next figure.

Message Authentication Code



Message Authentication Code

A MAC function is similar to encryption, but MAC algorithm is not necessarily reversible, and the checksum is of the fixed length.

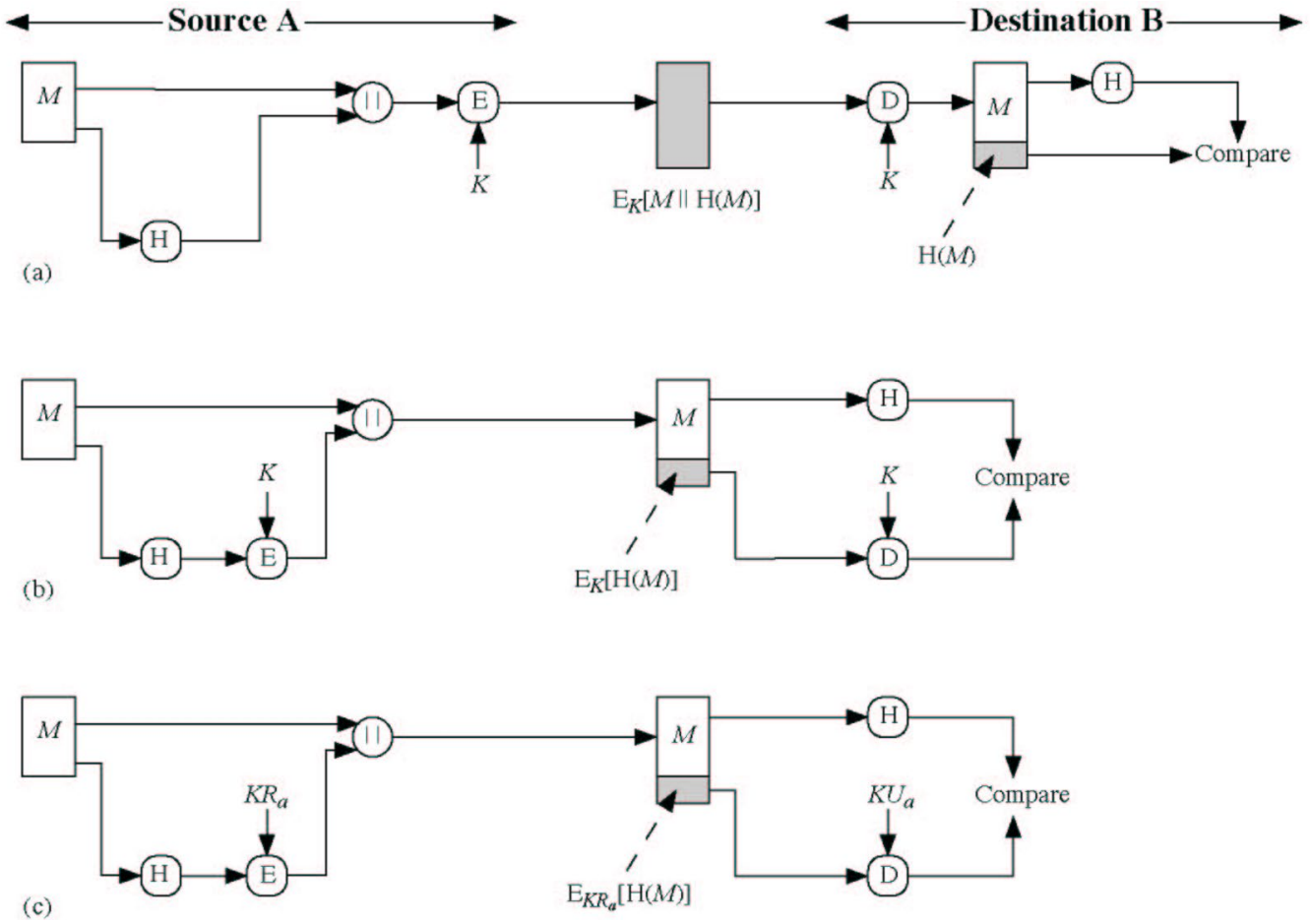
Advantages of using MAC rather than encryption:

- Authentication and confidentiality are separated - can be implemented on different levels
- It is faster - only MAC is encrypted
- Authentication can be checked on need basis
- Message is protected on the target machine and not only in transit

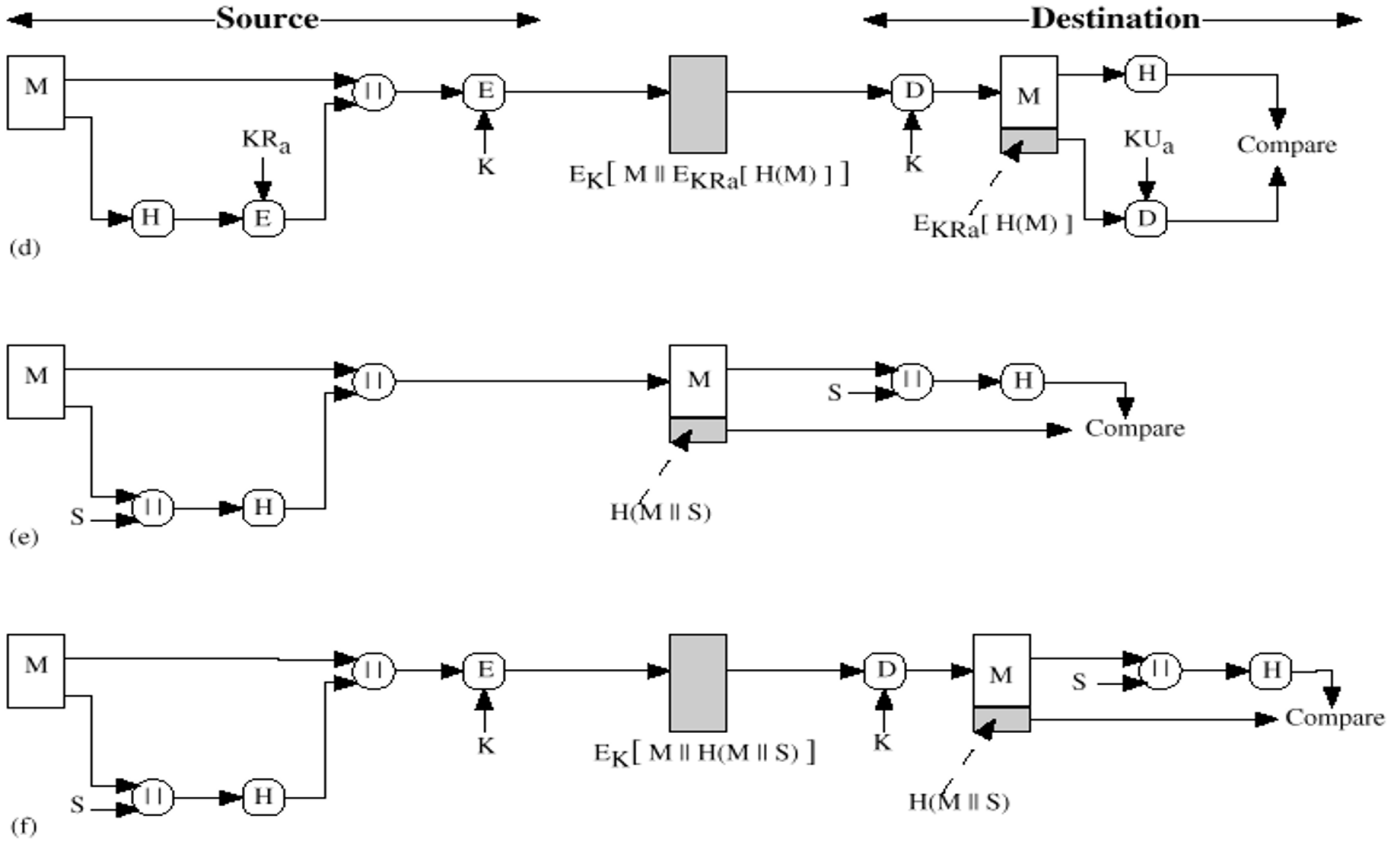
Hash Function

A hash function takes as input a variable-size message M and produces as output a fixed-size hash code $H(M)$ (also called a message digest). A hash function is one-way function - it is NOT reversible.

Different ways in which a hash code can provide authentication are illustrated in the next figure.



Hash Function



Reasons for avoiding encryption:

- Encryption is slow
- Hardware cost
- Hardware is optimized towards larger messages
- In the past some encryption algorithms were patented and were subjects to export controls

Message Authentication Codes

$$MAC = C_K(M)$$

MAC - fixed-length authenticator

M- variable-length message

K - secret key shared only by sender and receiver

Sender appends MAC to the message and sends it to the receiver. Receiver computes MAC and compares it with the received value.

In the case of encryption, the security generally depends on the size of the key - in a brute force attack, the opponent has to try on average 2^{k-1} different keys, where k is the number of bits in a key.

In the case of n-bit MAC, there are 2^n possible MACs and 2^k possible keys. Suppose that an opponent has intercepted some plaintext-MAC pairs, that is, M_i, MAC_i, where $MAC_i = C_K(M_1)$. The opponent can try all possible keys, but because $2^n < 2^k$, a number of keys (on average2^{k-n}) will produce a match.

Then opponent must repeat attack on another plaintext-MAC pair. Thus, brute-force in MAC is no less effort (maybe more) that the brute force in encryption with the key of the same length.

Message Authentication Codes

There are some attacks that do not require the discovery of the key.

Consider a message $M = (X_1||X_2|| \ldots ||X_m)$ where X_i is 128 bit block and let

$$\Delta(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_m \text{ and } C_K(M) = E_K[\Delta(M)]$$

where encryption algorithm is AES in electronic codebook mode.

The brute-force attempt to determine K requires at least 2^{128} encryptions. However, the opponent can construct a new message

$$Y = (Y_1||Y_2|| \ldots ||Y_m)$$

where Y_1 through Y_{m-1} are any 128 bit blocks and

$$Y_m = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1} \oplus \Delta(M)$$

and Y will have the same MAC as M.

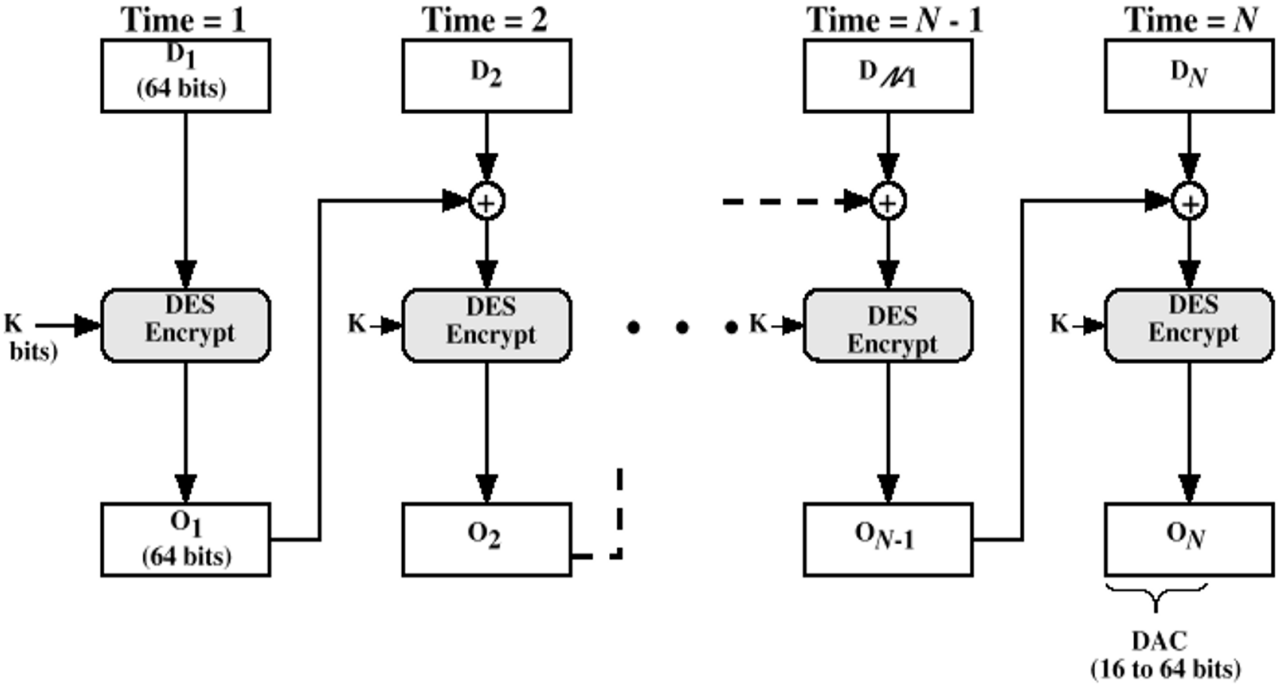
MAC Requirements

- It should be computationally infeasible to construct a message M' such that $C_K(M') = C_K(M)$, for a given message M.
- $C_K(M)$ should be uniformly distributed - for randomly chosen messages M and M' the probability that $C_K(M) = C_K(M')$ should be 2^{-n}.
- Let $M' = f(M)$; then again $\Pr[C_K(M) = C_K(M')] = 2^{-n}$

Data Authentication Algorithm

Data Authentication Algorithm was a NIST standard (**FIPS 113**) from 1985 to 2008. It is a MAC based on DES.

The message is split into 64 bit blocks $D_1, D_2, \ldots, D_N$. If the message length is not a multiple of 64, the last block is padded on the right with zeros. DAA uses Cipher Block Chaining mode of operation of DES.

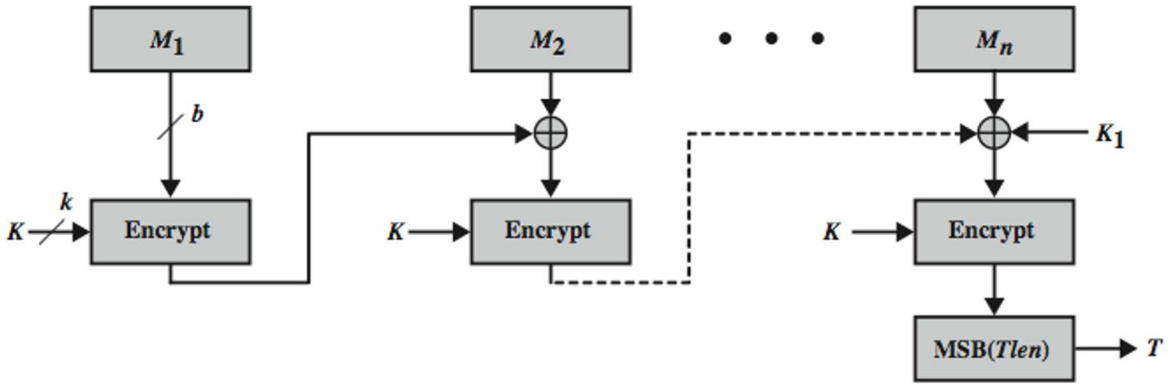


If DAA is used on a single block X, then an adversary who knows $T = MAC(K, X)$ also knows $MAC(X \,||\, (X \oplus T))$ as that is also equal to T.

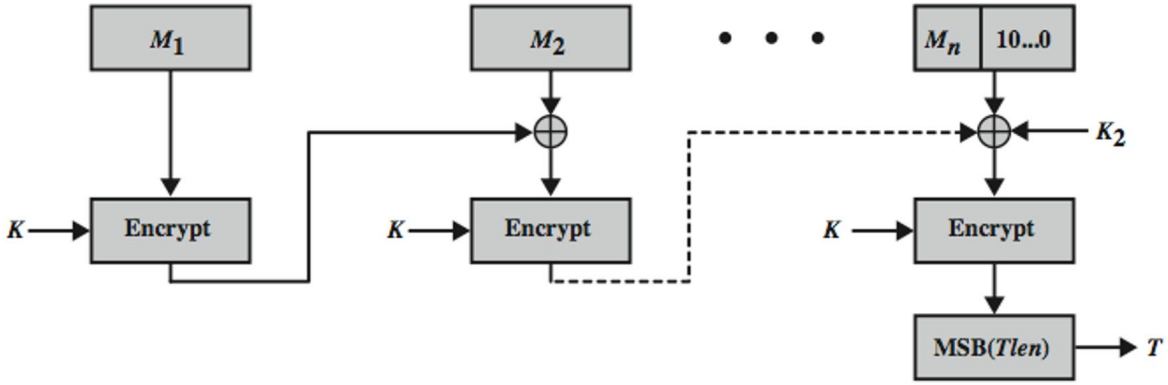
CMAC

At the time, DAA (CBC-MAC) was widely used by government and industry and it was considered to be secure, with the following restriction. Only messages of one fixed length of *mn* bits are processed, where *n* is the cipher block size and *m* is a fixed positive integer.

The weakness can be rectified by using 2 keys & padding, thus forming the Cipher-based Message Authentication Code (CMAC), adopted in 2005 by NIST SP800-38B for use with AES and triple DES, withdrawn in 2016.



(a) Message length is integer multiple of block size



CMAC Overview

CMAC uses the blocksize of the underlying cipher (ie 128-bits for AES or 64-bits for triple-DES).

The message is divided into n blocks $M_1..M_n$, padded if necessary by 1 followed by the required number of zeros.

The algorithm makes use of a k-bit encryption key K and an n-bit constant K_1 or K_2 (depending on whether the message was padded or not). For AES, the key size k is 128, 192 or 256 bits; for triple DES, the key size is 112 or 168 bits.

The two constants K_1 & K_2 are derived from the original key K using encryption of 0 and multiplication in $GF(2^n)$ as follows.

$$L = E(K,0^n)$$

$$K_1 = L \bullet x$$

$$K_2 = L \bullet x^2 = (L \bullet x) \bullet x$$

where $\bullet$ is a multiplication in $GF(2^n)$, and x is a first order polynomial in $GF(2^n)$, x = 00...010.

The irreducible polynomial for $GF(2^n)$ is the lexicographically first among all such polynomials with the minimum possible numbers of nonzero terms.

For n=64 and n=128 the irreducible polynomials are: $x^{64} + x^4 + x^3 + x + 1$ and $x^{128} + x^7 + x^2 + x + 1$

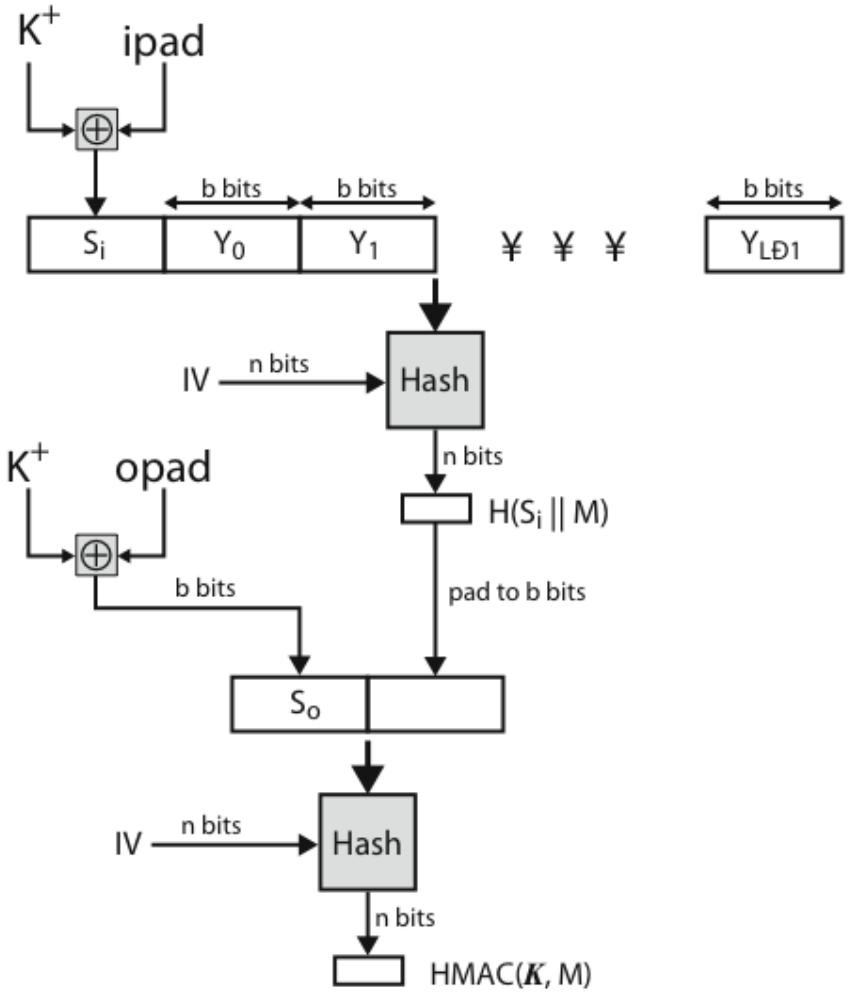
Keyed Hash Functions as MACs

- Advantages of MAC based on a hash function
  - hash functions are generally faster
  - codes for crypto hash functions are widely available
- Hash includes a key along with message.
- Original proposal:
  - `KeyedHash = Hash(Key|Message)`
  - some weaknesses were found with this
- This eventually led to development of HMAC.

HMAC Design Objectives

- It should use available hash functions without modification
- The embedded hash function should be easy to replace (e.g., if it becomes vulnerable, or if a faster one is available)
- It should not degrade the performance of the hash function
- It should handle keys in a simple way
- The cryptographic analysis should be well done and understood

HMAC Overview



HMAC Overview

We can describe the algorithm as follows:

1. Append zeros to the left end of *K* to create a *b*-bit string *K+*
2. XOR *K+* with ipad to produce the *b*-bit block *Si*.
3. Append M to Si.
4. Apply hash function H to the stream generated in step 3.
5. XOR *K+* with the opad to produce the *b*-bit block *S0*.
6. Append the hash result from step 4 to *S0*.
7. Apply H to the stream generated in step 6 and output the result.

- ipad – 36 in hexadecimal repeated *b*/8 times
- opad – 5C in hexadecimal repeated *b*/8 times

HMAC Security

- Designers of HMAC proved security of HMAC relates to that of the underlying hash algorithm
- Attacking HMAC requires either
  - brute force attack on key used, or
  - birthday attack (but since keyed would need to observe a very large number of messages).
- Choose hash function used based on speed verses security constraints

Next Week

1. Hash Functions
    a) Requirements
    b) Simple Hash Functions
    c) Birthday Attack
    d) Security
    e) Hash Algorithms: MD5 and SHA

2. Digital Signatures
    a) Direct Digital Signatures
    b) Arbitrated Digital Signatures
    c) Attacks and Forgeries
    d) Requirements
    e) Digital Signature Algorithms

Chapter 11 from text: Cryptographic Hash Functions
Chapter 13 from text: Digital Signatures

References

1. W. Stallings. "*Cryptography and Network Security*", Global Edition, Pearson Education, 2017.

2. W. Stallings, "Cryptography and Network Security" Official Slides.

3. L. Brown "Cryptography and Network Security" Lecture Slides accompanying the textbook.