Network Security
Public-key Crypto

Amir Rezapour
Institute of Information Security,
National Tsing Hua University

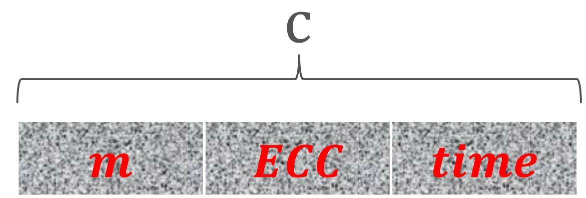
Approaches to Message Authentication

- Encryption protects against passive attack.
- How to protect messages against active attack?
  - Modification and falsification of messages
- A message is *authentic* if
  - The contents of the message have not been altered
  - The message indeed comes from the claimed source.

Approaches to Message Authentication

- Using conventional encryption
  - We assume that only the sender and receiver share a key, so only the genuine sender would be able to encrypt a message successfully.
  - The receiver assumes that no alterations have been made and that sequencing is proper if the message includes an error detection code and a sequence number
  - If the message includes a timestamp, the receiver assumes that the message has not been delayed beyond that normally expected for network transit

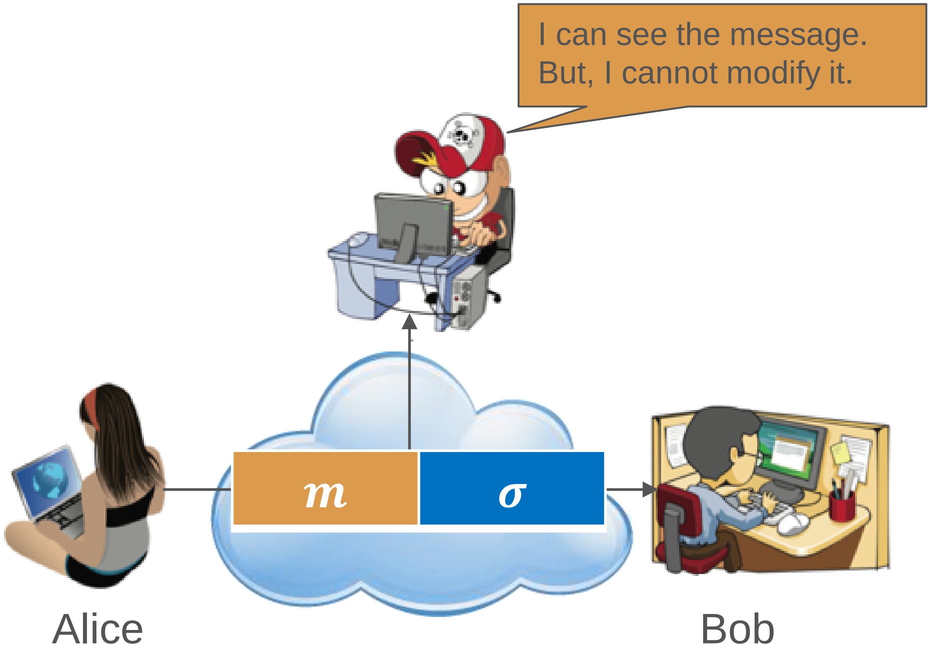
Approaches to Message Authentication



- Why not symmetric encryption?
  - Alice and Bob share a secret key.
    - Assuming that Bob can recognize invalid messages, only Alice can encrypt a message for Bob.
  - If the message includes an error-detection code, then Bob is confident that no modifications have been made during message transmission.
  - If the message contains a timestamp, then Bob is confident that the message has not been delayed.
- BUT, symmetric encryption is *not* a suitable tool!
  - E.g., if Alice and Bob use ECB mode of operation
    - An attacker can reorder the blocks of ciphertext.

Approaches to Message Authentication

- Without message encryption
  - An authentication *tag* is generated and appended to each message for transmission
  - Bob use the tag to verify the authenticity.

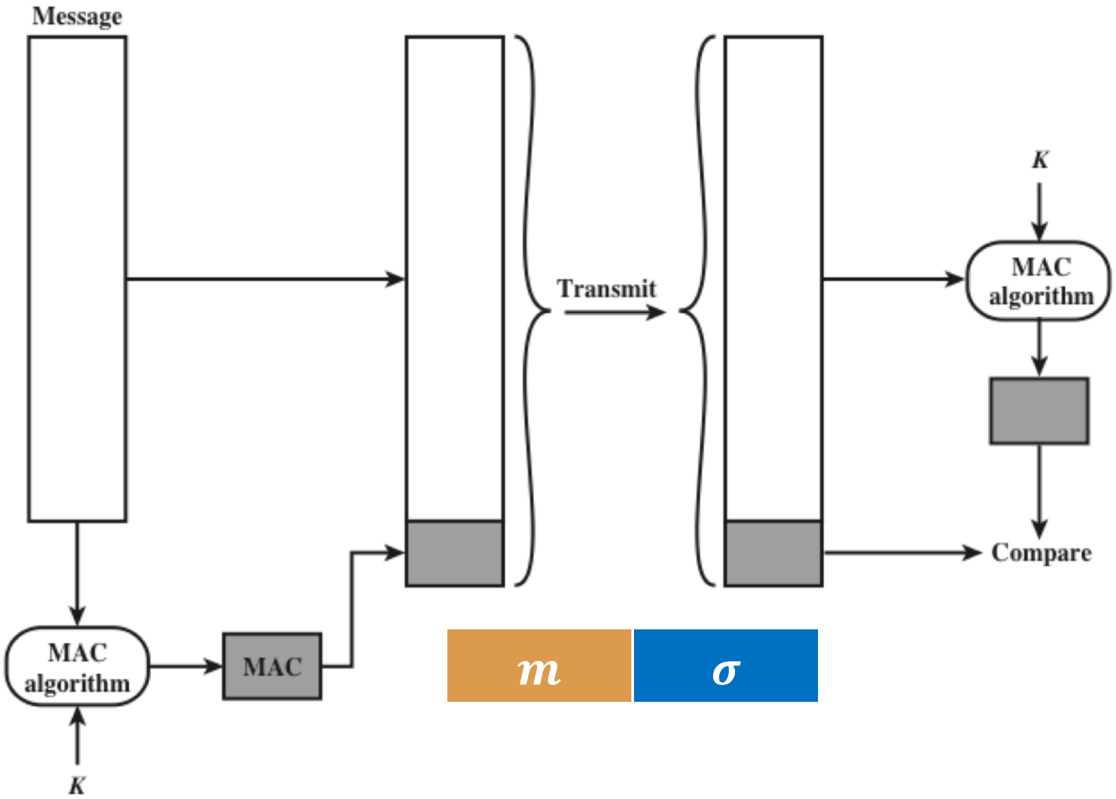


Approaches to Message Authentication

- Why just authentication tag without encryption?
  - Suppose you broadcast a signal to the servers within the company that some network/recourse is unavailable.
  - Suppose the load of a webservice is high and it cannot afford to decrypt all incoming messages.
    - Maybe choose some messages randomly for checking.
  - Authentication of a software in plaintext is more attractive.
    - No need to decrypt it every time.
    - It just checks the integrity of the software.

Approaches to Message Authentication

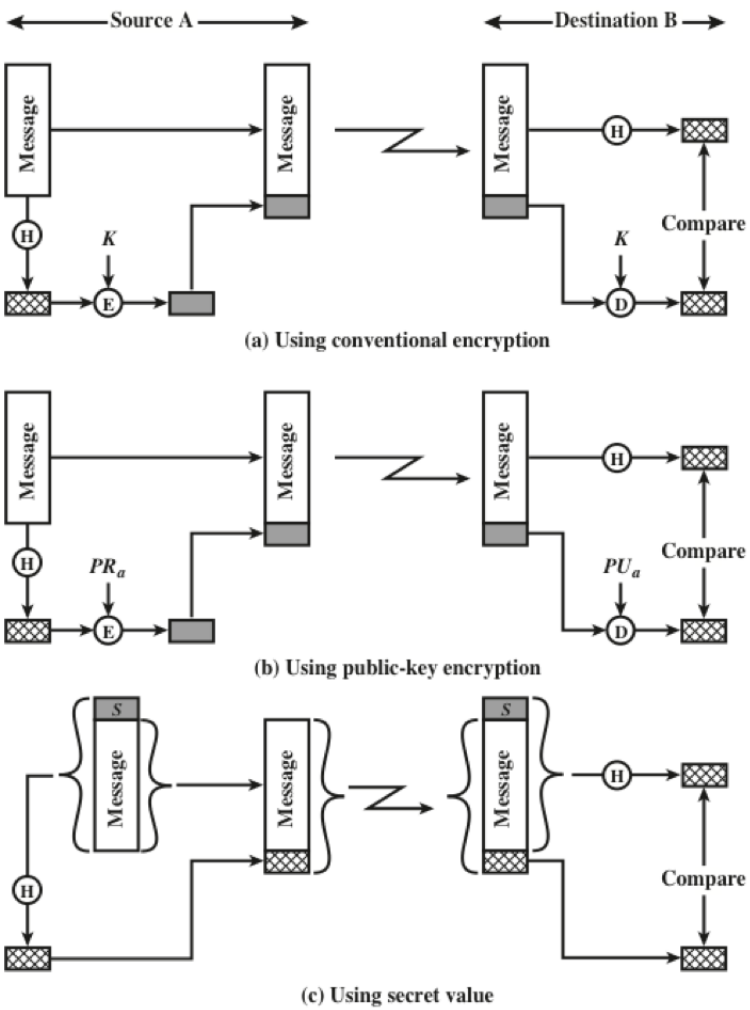
- Message Authentication Code (MAC)
  - Shared key K
  - $\sigma = MAC(K, M)$



One-way Hash Functions

- $H: \{0,1\}^* \to \{0,1\}^n$
  - Accepts a variable-size message M as input
  - Produces a fixed-size digest $|H(M)| = n$ as output
- Does not take a secret key as input
- To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic

One-way Hash Functions



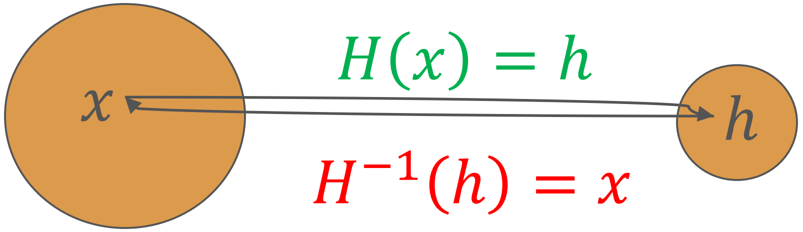
Secure Hash Functions

1) H can be applied to a block of data of any size.
2) H produces a fixed-length output.
3) $H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.

- These are the requirements for a practical hash function.

Secure Hash Functions

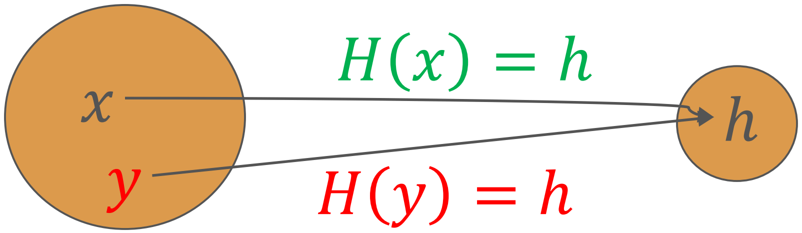
4) For any given code h, it is computationally infeasible to find x such that $H(x) = h$.
   - A hash function with this property is referred to as one-way or *preimage resistant*.



   - One-way is important.
     - In HMAC, $\sigma = H(K||M)$.
     - If HMAC is ***not*** one-way, then attacker inverts it $K||M = H^{-1}(\sigma)$.

Secure Hash Functions

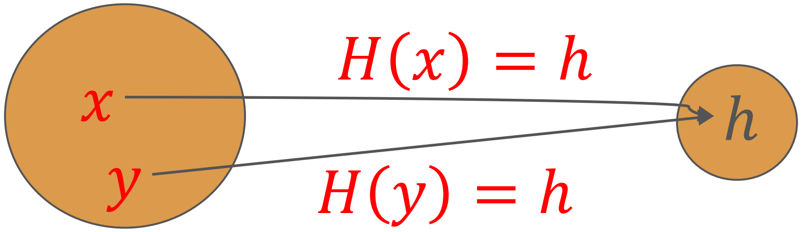
5) For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
   – A hash function with this property is referred to as second preimage resistant. This is sometimes referred to as *weak collision resistant*.



- This prevents forgery.

Secure Hash Functions

6) It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
   - A hash function with this property is referred to as *collision resistant*. This is sometimes referred to as *strong collision resistant*.



Security of Hash Functions

- There are two approaches to attacking a secure hash function:
  - Cryptanalysis
    - Involves exploiting logical weaknesses in the algorithm
  - Brute-force attack
    - The strength of a hash function against this attack depends solely on the length of the hash code produced by the algorithm.
    - MD5 with $n = 128$ is inadequate.
    - We need $n \geq 160$.

Preimage resistant	$\mathbf{2^n}$
Second preimage resistant	$\mathbf{2^n}$
Collision resistant	$\mathbf{2^{\frac{n}{2}}}$

Simple Hash Function 1

- $C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$
  - b_{ij} is ith bit in jth block.

	bit 1	bit 2	• • •	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

Simple Hash Function 2

- Suppose a message is chopped into blocks of $M_1, M_2, \dots, M_N$

1) The Hash Code $C = M_{N+1} = M_1 \oplus M_2 \oplus \dots \oplus M_N$
2) Then $cbcE(K, M_1, M_2, \dots, M_N, M_{N+1}) = Y_1, Y_2, \dots Y_N, Y_{N+1}$

- It does not work!
  - Recall the CBC mode
    - $M_1 = IV \oplus D(K, Y_1)$
    - $M_i = Y_{i-1} \oplus D(K, Y_i)$
    - $M_{N+1} = Y_N \oplus D(K, Y_{N+1})$
  - The hash code M_{N+1} cannot detect if ciphertext were permuted!

$$M_{N+1} = M_1 \oplus M_2 \oplus \dots \oplus M_N$$
$$= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_1)] \oplus \dots \oplus [Y_N \oplus D(K, Y_N)]$$

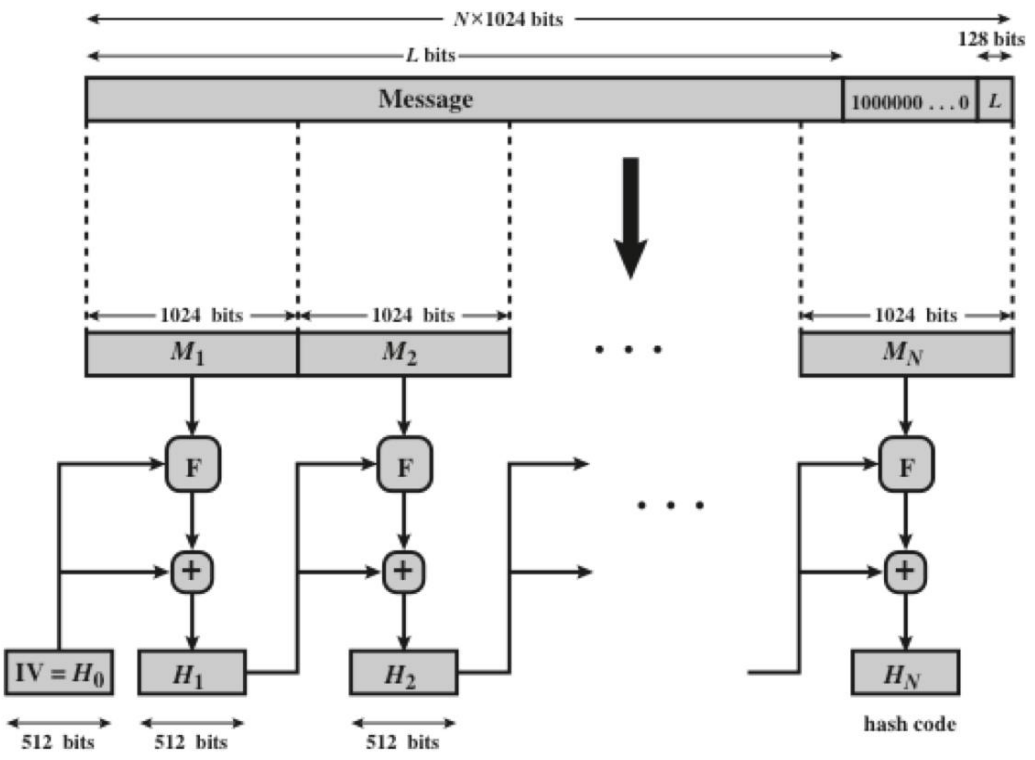
The SHA Secure Hash function

- SHA was developed by NIST and published in 1993
- The actual standards document is entitled “Secure Hash Standard”
- SHA-1 produces 160-bit hash values
- In 2005 NIST announced the intention to phase out approval of SHA-1 and move to a reliance on SHA-2 by 2010

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

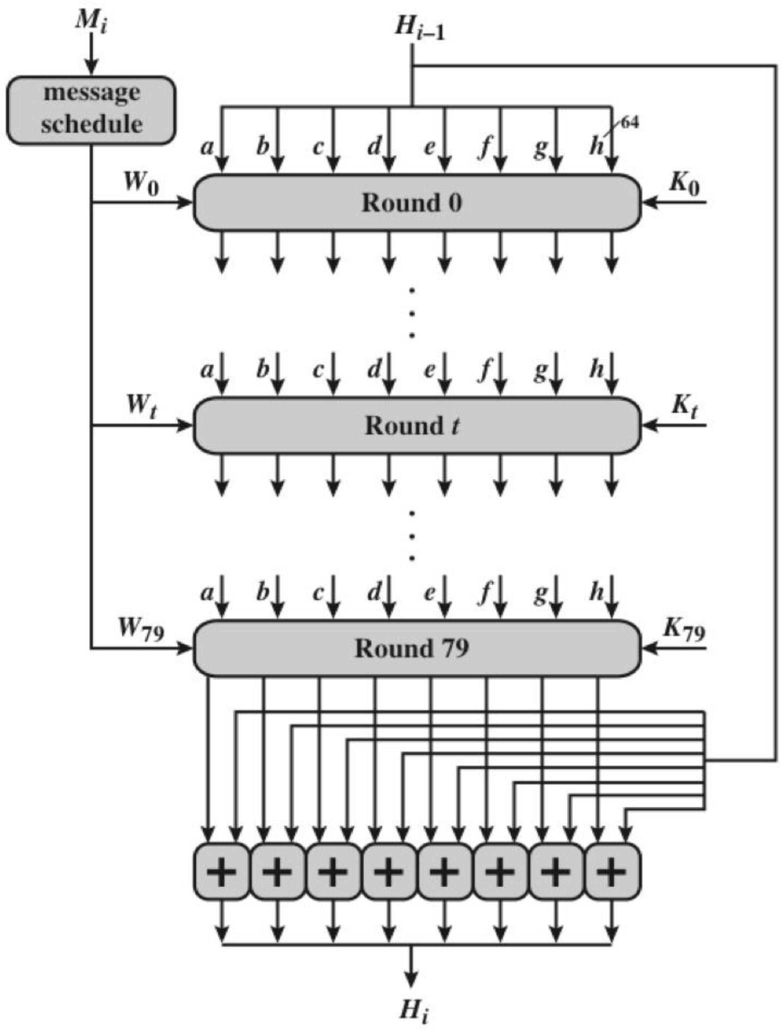
The SHA Secure Hash function

- Append length
  - $L = |M|$ before padding
- Padding
  - Add $1\{0\}^*$ until to M
    - $|M| \equiv 896 \; mod \; 1024$



The SHA Secure Hash function

$$a = 0x6a09e667f3bcc908,$$
$$b = 0xbb67ae8584caa73b$$
$$c = 0x3c6ef372fe94f82b$$
$$d = 0xa54ff53a5f1d36f1$$
$$e = 0x510e527fade682d1$$
$$f = 0x9b05688c2b3e6c1f$$
$$g = 0x1f83d9abfb41bd6b$$
$$h = 0x5be0cd19137e2179$$



SHA-3

1. It must be possible to replace SHA-2 with SHA-3 in any application by a simple drop-in substitution. Therefore, SHA-3 must support hash value lengths of 224, 256, 384, and 512 bits.

2. SHA-3 must preserve the online nature of SHA-2. That is, the algorithm must process comparatively small blocks (512 or 1024 bits) at a time instead of requiring that the entire message be buffered in memory before processing it.

Basic requirements that must be satisfied by any candidate for SHA-3

HMAC

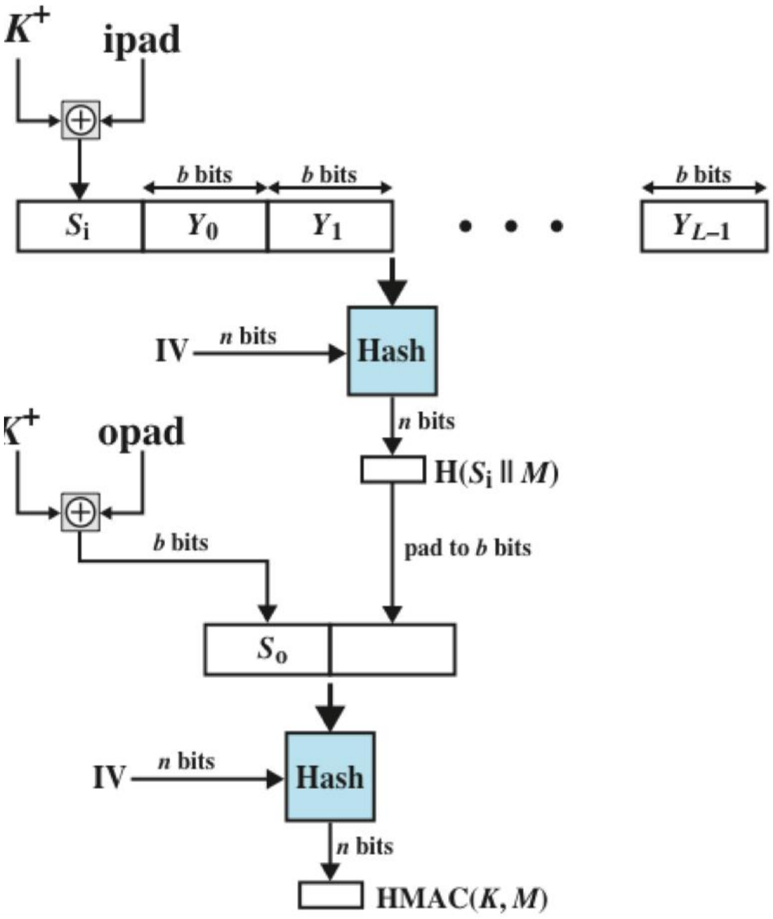
- There has been an increased interest in developing a MAC derived from a cryptographic hash code, such as SHA-1
  - Cryptographic hash functions generally execute faster in software than conventional encryption algorithms such as DES
  - A hash function such as SHA-1 was not designed for use as a MAC and cannot be used directly for that purpose because it does not rely on a secret key

HMAC Design Objectives

- To use an available hash function as a *Blackbox* without modifications
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required
- To preserve the original performance of the hash function without incurring a significant degradation
- To use and handle keys in a simple way

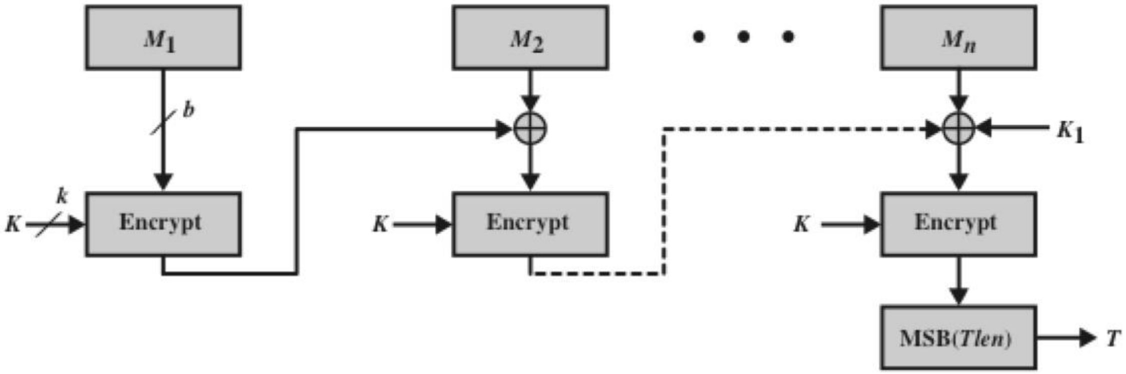
HMAC

- H is for example SHA-2
- K is secret key
  - If $|K| > b$, then $K = H(K)$
- K^+ is K padded with $\{0\}^*$
  - Until $|K| = b$
- ipda and opad are constants
  - They make 2 different keys
  - by flipping half of k^+'s bits



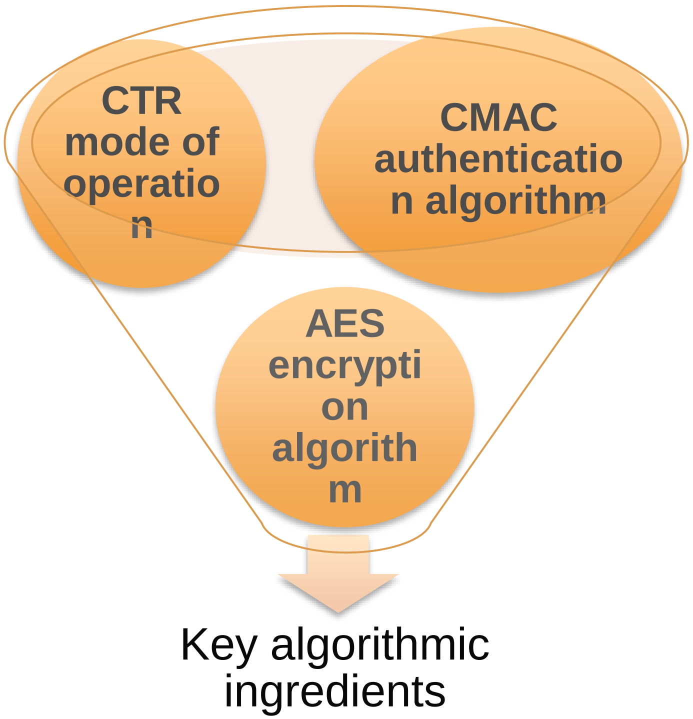
CMAC

- T is MAC (a.k.a. tag)
- $Tlen$ is length of T
- $T = MSB_{Tlen}(C_n)$
  - $Tlen$ most significant bits of C_n

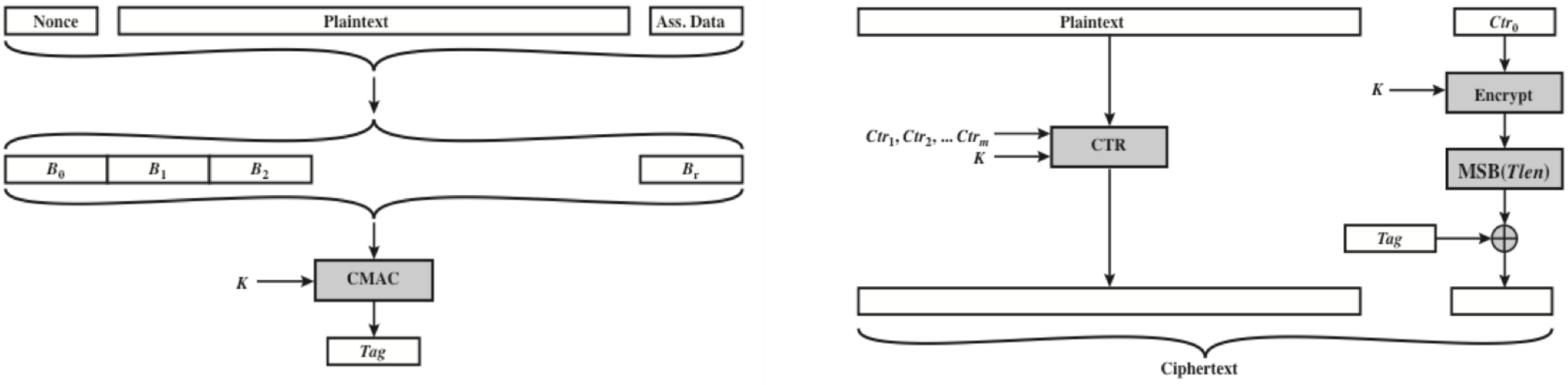


Counter with Cipher Block Chaining-Message Authentication Code (CCM)

- Referred to as an authenticated encryption mode
  - protect confidentiality and authenticity of communications
- A single key is used for both encryption and MAC algorithms



Counter with Cipher Block Chaining-Message Authentication Code (CCM)



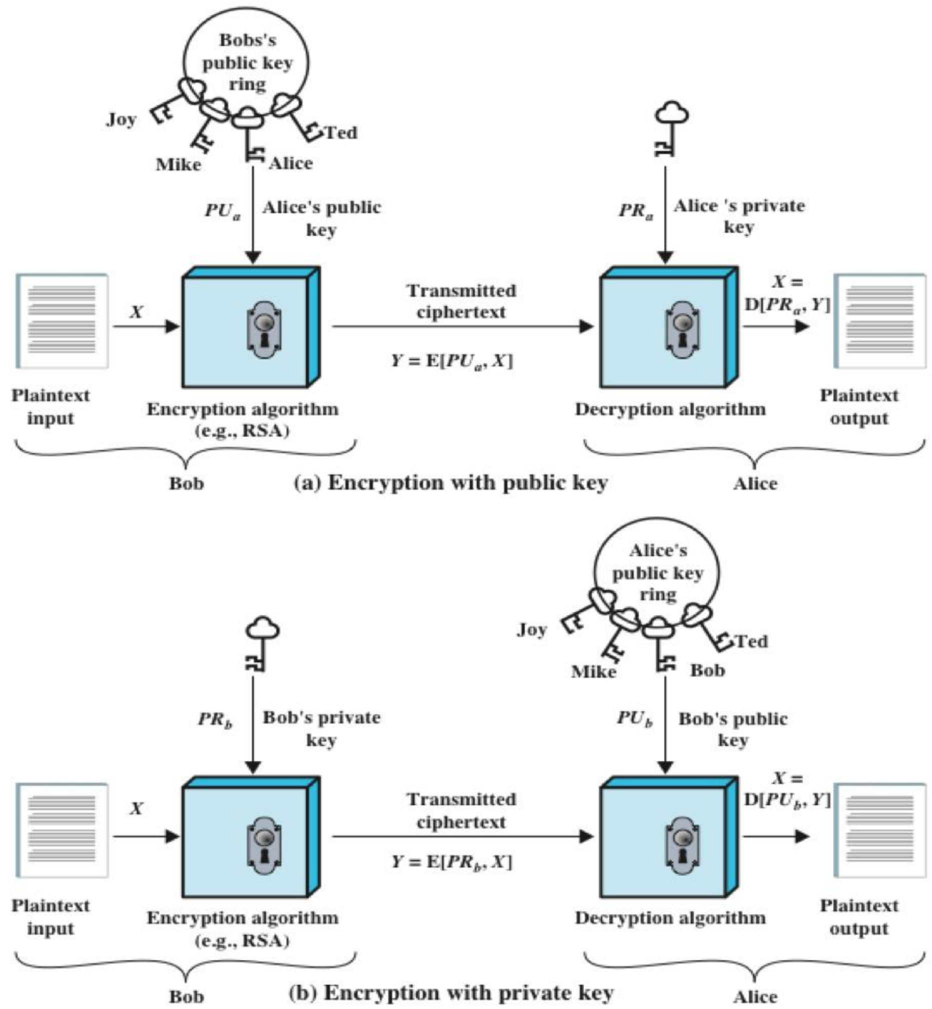
(a) Authentication

(b) Encryption

Public-Key encryption structure

- First publicly proposed by Diffie and Hellman in 1976
- Based on mathematical functions rather than on simple operations on bit patterns
- Is asymmetric, involving the use of two separate keys
- Public-key encryption is more secure from cryptanalysis than conventional encryption
- There is a feeling that key distribution is trivial when using public-key encryption

Public-Key encryption structure



Applications for public-key cryptosystems

The use of public-key cryptosystems can be classified into three categories:

- Encryption/decryption — The sender encrypts a message with the recipient’s public key
- Digital signature — The sender “signs” a message with its private key
- Key exchange — Two sides cooperate to exchange a session key

Applications for public-key cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No
Elliptic Curve	Yes	Yes	Yes

Public key requirements

1. It is computationally easy for Bob to generate public and private key pair (PU_B, PR_B).
2. It is computationally easy for a sender Alice, to compute $C = E(PU_B, M)$
3. It is computationally easy for the recipient Bob to obtain the original message $M = D(PR_B, C)$
4. It is computationally infeasible for an attacker to obtain PR_B from PU_B
5. It is computationally infeasible for an attacker knowing PU_B and C, to recover M

RSA

- One of the first public-key schemes developed in 1977
- It is proposed by **R**ivest, **S**hamir, and **A**dleman
- Based on the idea that factorization of integers into their prime factors is hard.
  - Given n, it is hard to find p and q s.t. $\mathrm{n} = p \times q$
  - E.g., $n = 15$, $p = 3$ and $q = 5$.
  - How about

$$n = 2610121787199409810684117643702615348860619383579124216801775762670146114509129578115653691483544207585509866476058959961$$

RSA

- Parameters
  - $n = p \times q$, $|n| = 1024$-bits , $|p|, |q| \approx 512$-bits
- KeyGen()
  - $\phi(n) = (p-1).(q-1)$
  - Randomly choose e s.t. $\gcd(e, \phi(n)) = 1$
  - Calculate d from $ed = 1\ mod\ \phi(n)$
  - Keep $PR = (d, n)$ as secret and publish $PU = (e, n)$.
- Encrypt
  - For a message $M \in \{0,1, \ldots, n-1\}$
  - $C = M^e\ mod\ n$
- Decrypt
  - $M = C^d\ mod\ n$

RSA

- Correctness
  - Since $\gcd\big(e, \phi(n)\big) = 1$, there exists a $k \in \mathbb{N}$, s.t.
  - $ed = k.\phi(n) + 1$
  - By the Euler's Totient Theorem
    - If $\gcd\big(a, \phi(n)\big) = 1$, $a^{\phi(n)} = 1\ mod\ n$

$$C^d = M^{ed} = M^{k.\phi(n)+1} = M\ mod\ n$$

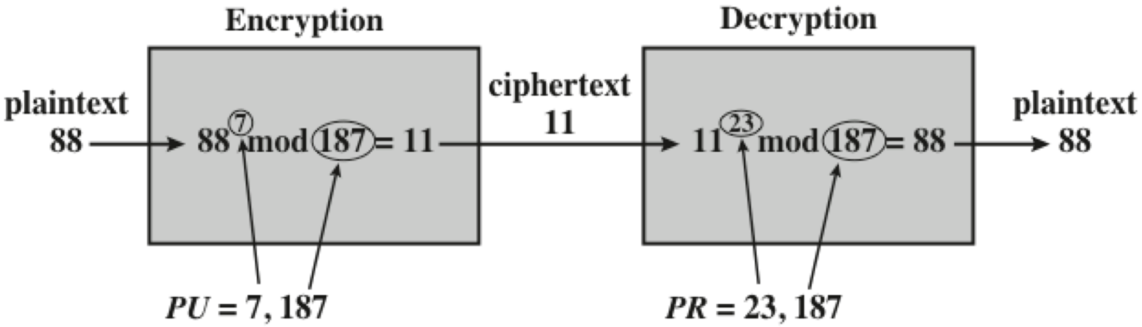
- Security
  - Factorization is hard.
  - A 1024 bit n is 300 decimal digits.

RSA

- Security
  - The values m = 0 or m = 1 always produce ciphertexts equal to 0 or 1 respectively
  - When e and m are small (e.g., $e = 3$) the (non-modular) result of $m^e < \mathrm{n}$.
    - Then the cyphertext can be decrypted easily by taking the eth root of the ciphertext with no regard to the modulus.
  - RSA encryption is a deterministic encryption algorithm. It has no random component.
    - Attacker can run chosen plaintext attack.

RSA: Example

- Parameters
  - Let $p = 17$ and $p = 11$, then $\mathrm{n} = 17 \times 11 = 187$
- KeyGen()
  - $\phi(n) = (p-1).(q-1) = 16 \times 10 = 160$
  - Randomly choose $e = 7$ s.t. $\gcd\big(7, \phi(n)\big) = 1$
  - Calculate d from $ed = 1\ mod\ \phi(n)$
    - $d = 23$, because $23 \times 7 = 161 = 1\ mod\ 160$



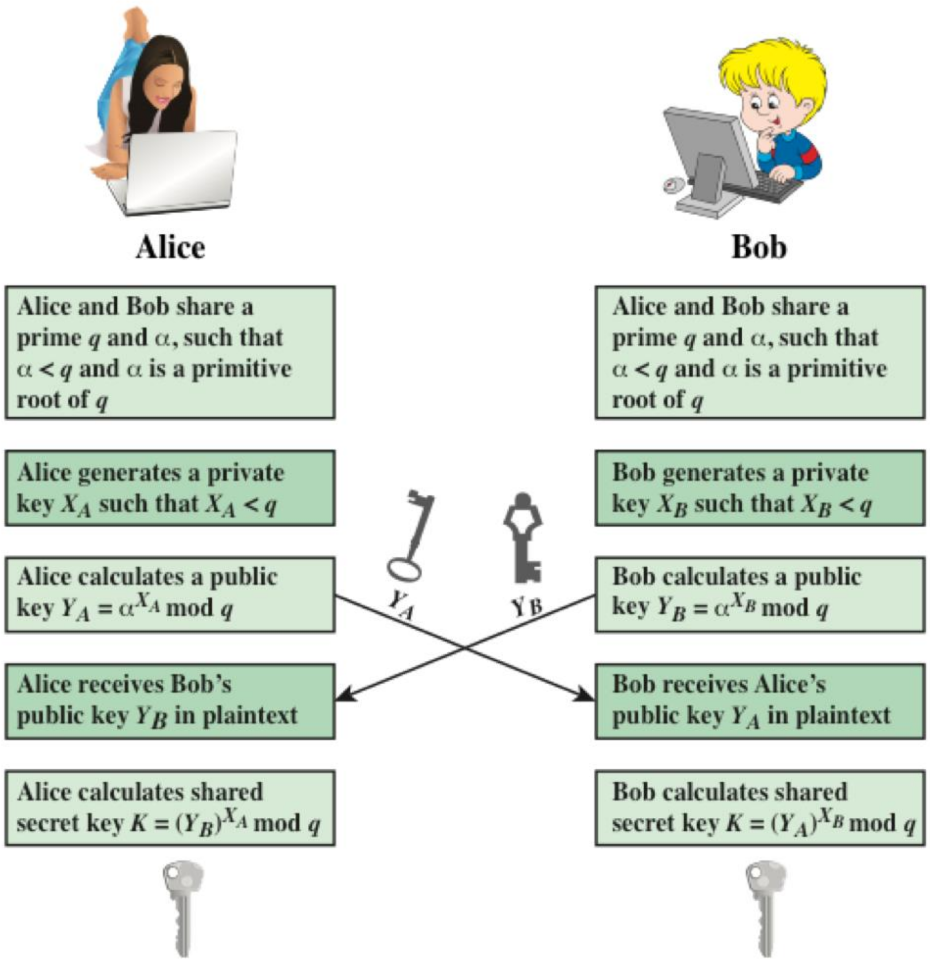
RSA Signature

- Suppose Alice $PR = (d, n)$ want to sign a message m
- Sign(m, PR)
  - Compute $h = H(m)$
  - $\sigma = h^d \, mod \, n$
  - Send m, σ to Bob
- Verify(m, σ)
  - Compute $h' = H(m)$
  - $k = \sigma^e \, mod \, n$
  - Check if $h = h'$?

Diffie-Hellman Key Exchange

- First published public-key algorithm
- A number of commercial products employ this key exchange technique
- Purpose of the algorithm is to enable two users to exchange a secret key securely that then can be used for subsequent encryption of messages
- The algorithm itself is limited to the exchange of the keys
- Depends for its effectiveness on the difficulty of computing discrete logarithms

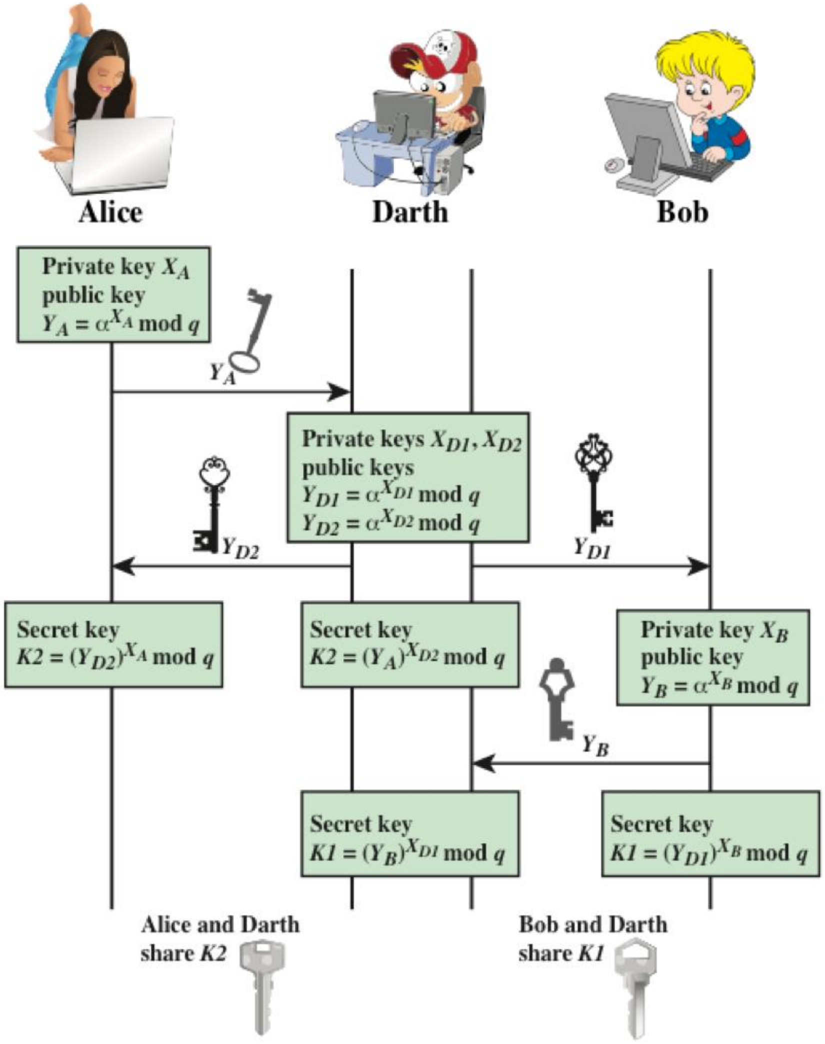
Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange

- Correctness
  - Alice
    - $K = Y_B^{X_A} = (\alpha^{X_B})^{X_A}\ mod\ q$
  - Bob
    - $K = Y_A^{X_B} = (\alpha^{X_A})^{X_B} mod\ q$
- Security
  - Given a prime q and its primitive root α, and Y_A, it is computationally infeasible to find X_A s.t. $Y_A = \alpha^{X_A}\ mod\ q$

Man-in-the-Middle Attack



Digital Signature standard (DSS)

- Makes use of the SHA-1 and presents a new digital signature technique, the Digital Signature Algorithm (DSA)
- Originally proposed in 1991 and revised in 1993 and again in 1996
- Uses an algorithm that is designed to provide only the digital signature function
- Unlike RSA, it cannot be used for encryption or key exchange

Elliptic-curve cryptology (ECC)

- Technique is based on the use of a mathematical construct known as the elliptic curve
  - Principal attraction of ECC compared to RSA is that it appears to offer equal security for a far smaller bit size, thereby reducing processing overhead
    - For the level of security that can be achieved by an elliptic curve cryptography key of 256 bit requires an RSA key to be 3072 bit.
- The confidence level in ECC is not yet as high as that in RSA

Summary

- Approaches to message authentication
  - Authentication using conventional encryption
  - Message authentication without message encryption
- Secure hash functions
  - Hash function requirements
  - Security of hash functions
  - Simple hash functions
  - The SHA secure hash function SHA-3
- Digital signatures
- Message authentication codes
  - HMAC
  - MACs based on block ciphers
- Public-key cryptography principles
  - Public-key encryption structure
  - Applications for public-key cryptosystems
  - Requirements for public-key cryptography
- Public-key cryptography algorithms
  - The RSA public-key encryption algorithm
  - Diffie-Hellman key exchange
  - Other public-key cryptography algorithms