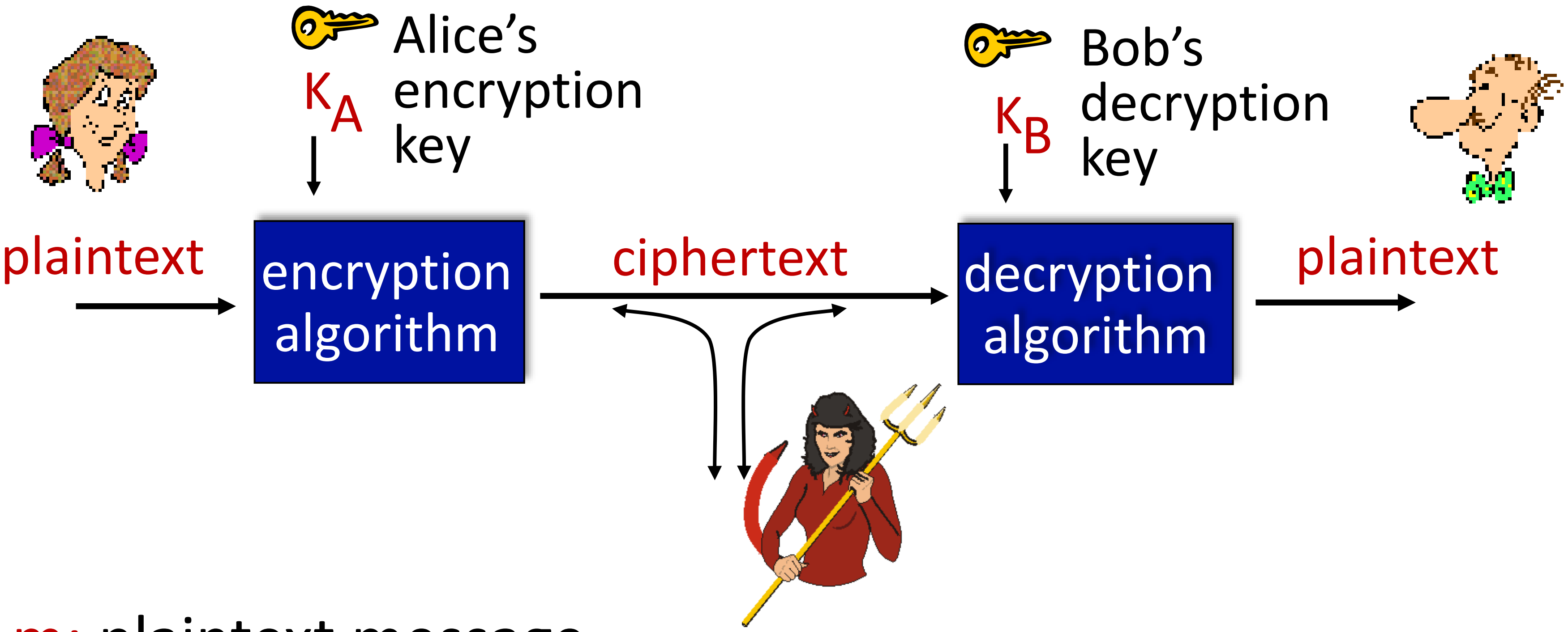
Advanced Network Security

Amir Mahdi Sadeghzadeh, Ph.D.

Outline

- What is network security?
- Principles of cryptography
- Message integrity, authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec

The language of cryptography

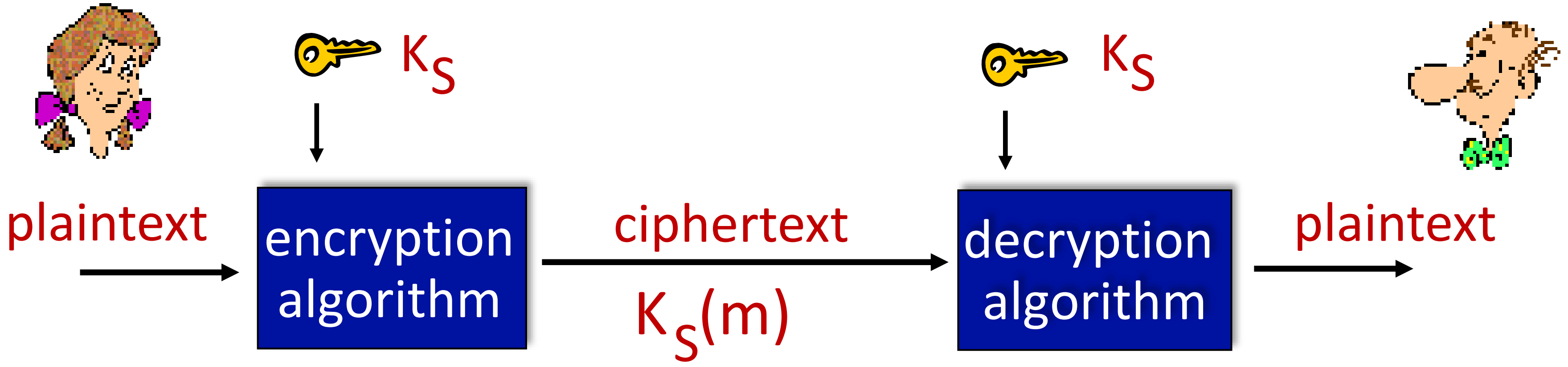


m: plaintext message

$K_A(m)$: ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K

- *e.g.*, key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

Public Key Cryptography

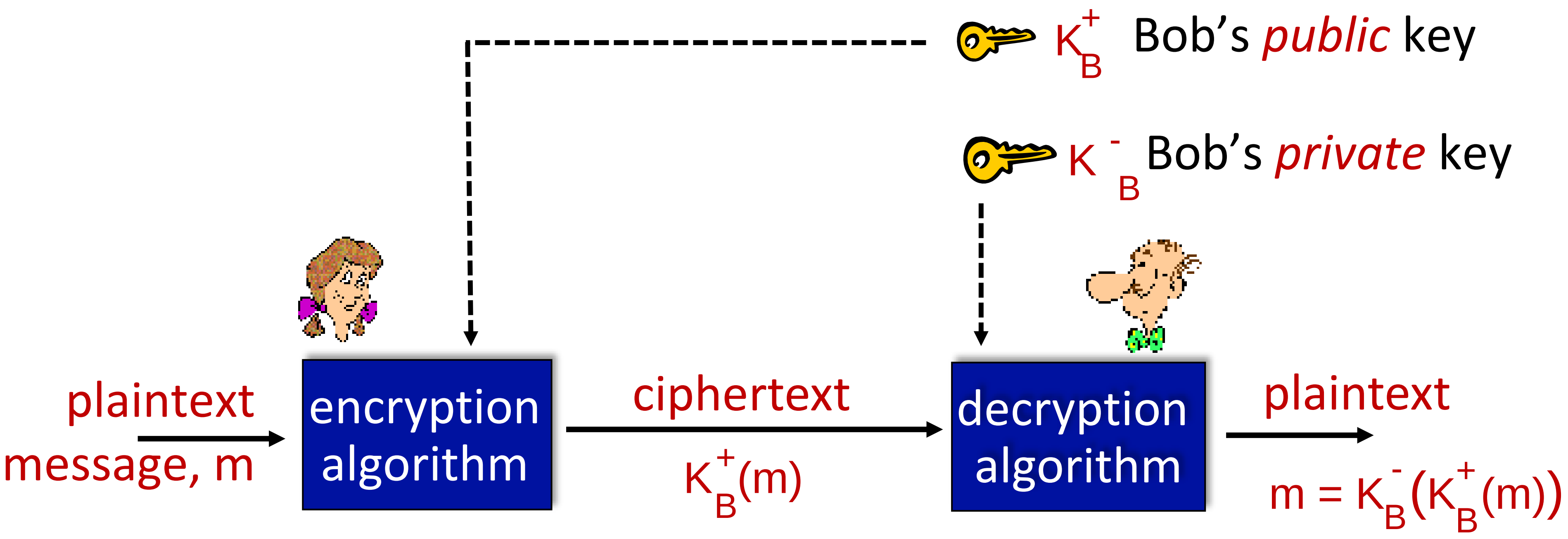
symmetric key crypto:

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?
- Message integrity, authentication?

public key crypto

- *radically* different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public Key Cryptography



Public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

- similar ideas emerged at roughly same time, independently in US and UK (classified)

Public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

RSA: Creating public/private key pair

1. choose two large prime numbers *p, q.* (e.g., 1024 bits each)

2. compute $n = pq$, $\phi(n) = z = (p-1)(q-1)$

3. choose *e* (with $e<z$) that has no common factors with z (*e, z* are "relatively prime"). The number e is usually 65537 (0x010001).

4. choose *d* such that $ed-1$ is exactly divisible by *z*. (in other words: $ed \bmod z = 1$).

5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above

1. to encrypt message m ($<n$), compute

   $c = m^e \bmod n$

2. to decrypt received bit pattern, c, compute

   $m = c^d \bmod n$

$$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$$

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

result is the same!

Homomorphic Property

- In mathematics and cryptography, a homomorphism is a function that preserves certain algebraic operations.
  - RSA exhibits a homomorphic property with respect to multiplication.
  - If you encrypt two plaintexts and then multiply their ciphertexts, the result is equivalent to encrypting the product of the plaintexts.

  - $m_3 = m_1 . m_2$
  - $c_3 = \ m_3^e \ mod \ n = (m_1 . m_2)^e \ mod \ n = m_1^e . m_2^e \ mod \ n$
  - $= (m_1^e \ mod \ n . m_2^e \ mod \ n) \ mod \ n = (c_1 . c_2) \ mod \ n$
  - $c_3 = c_1 . c_2 \ mod \ n$

Chapter 8 outline

- What is network security?
- Principles of cryptography
- **Authentication, message integrity**
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec

Other than Confidentiality

- Authentication
  - Definition: The process of verifying the identity of a user, device, or system.
    - Ensures that the entity accessing the system is who they claim to be.
  - Methods: Passwords, biometrics, digital signatures, MACs, etc.
- Data Integrity
  - Definition: The assurance that data remains unaltered during transmission or storage.
  - Methods: MACs, digital signatures.
- Non-repudiation
  - Definition: The inability of a user to deny the authenticity or origin of a communication or action.
  - Methods: Digital signatures.

Where Does This Fit?

	Secret Key Setting	Public Key Setting
Secrecy / Confidentiality	Stream cipher Block cipher + encryption modes	Public key encryption: RSA, El Gamal, etc.
Authenticity / Integrity	MAC	Digital Signatures

Introduction to Cryptography CS 355, Purdue

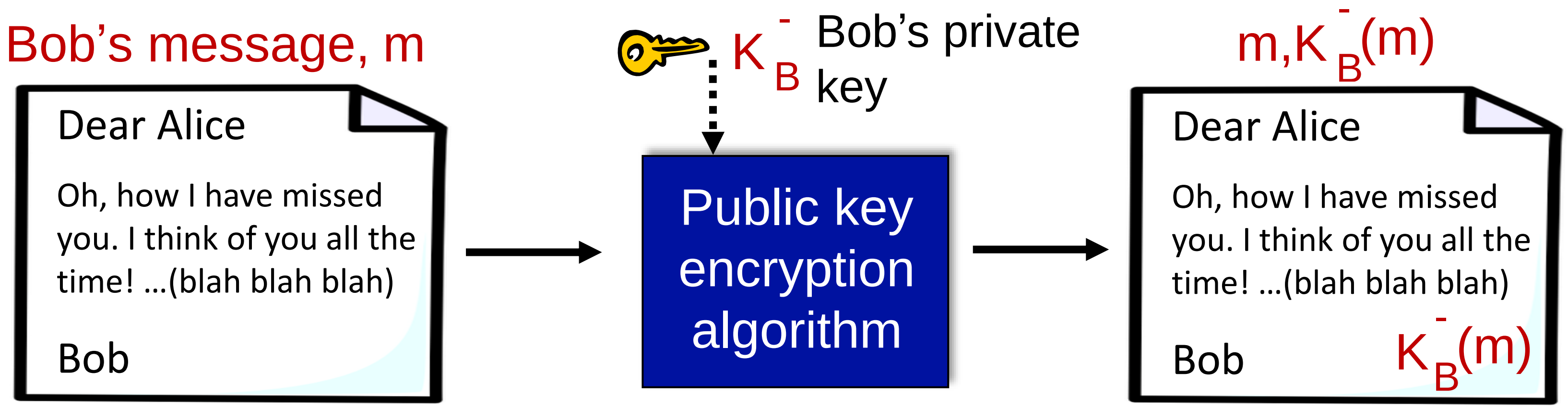
Digital Signatures: The Problem

- Consider the real-life example where a person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
- Contracts, they are valid if they are signed.
- Can we have a similar service in the electronic world?

Digital signatures

Cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document: he is document owner/creator.
- *verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
- simple digital signature for message m:
  - Bob signs m by encrypting with his private key K_B, creating "signed" message, $K_B^-(m)$



Digital signatures

- suppose Alice receives msg m, with signature: m, $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key

Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

Security properties of digital signature

- Message authentication
  - When the verifier validates the digital signature using public key of a sender, he is assured that **signature has been created only by sender who possess the corresponding secret private key** and no one else.

- Data Integrity
  - In case **an attacker has access to the data and modifies it, the digital signature verification at receiver end fails**. The modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.

- Non-repudiation
  - Since it is assumed that **only the signer has the knowledge of the signature key**, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

Attack Models for Digital Signatures

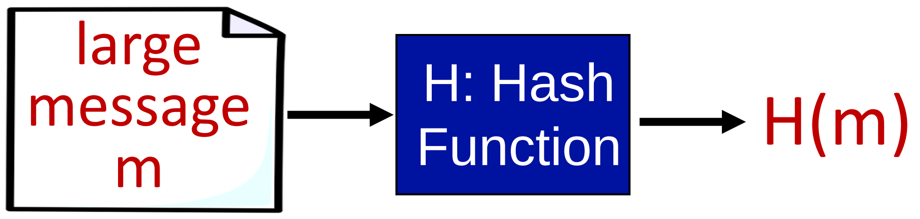
- Key-only attack
  - Adversary knows only the verification function, including **victim's public key** (which is supposed to be public).
- Known message attack
  - Adversary knows **a list of messages previously signed** by victim.
- Chosen message attack
  - Adversary can **choose what messages wants victim to sign**, and he knows both the messages and the corresponding signatures.

Adversarial Goals

- Total break
  - adversary is able **to find the secret (victim's private key)** for signing, so he can forge then any signature on any message.
- Selective forgery
  - adversary is able to **create valid signatures on a message chosen** by someone else, with a significant probability.
- Existential forgery
  - **adversary can create a pair (message, signature)**, s.t. the signature of the message is valid.
    - Inverse public key process
    - Homomorphic property

Message digests

- Computationally expensive to public-key-encrypt long messages
  - fixed-length, easy- to-compute digital “fingerprint”
  - apply hash function H to *m*, get fixed size message digest, *H(m)*
- Existential forgery
  - extra layer of security



Hash function properties:

- produces fixed-size msg digest (fingerprint)
- given message digest *x*, computationally infeasible to find *m* such that $x = H(m)$

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function

- produces fixed length digest (16-bit sum) of message

but given message with given hash value, it is easy to find another message with same hash value:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

message	ASCII format
I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42
	B2 C1 D2 AC

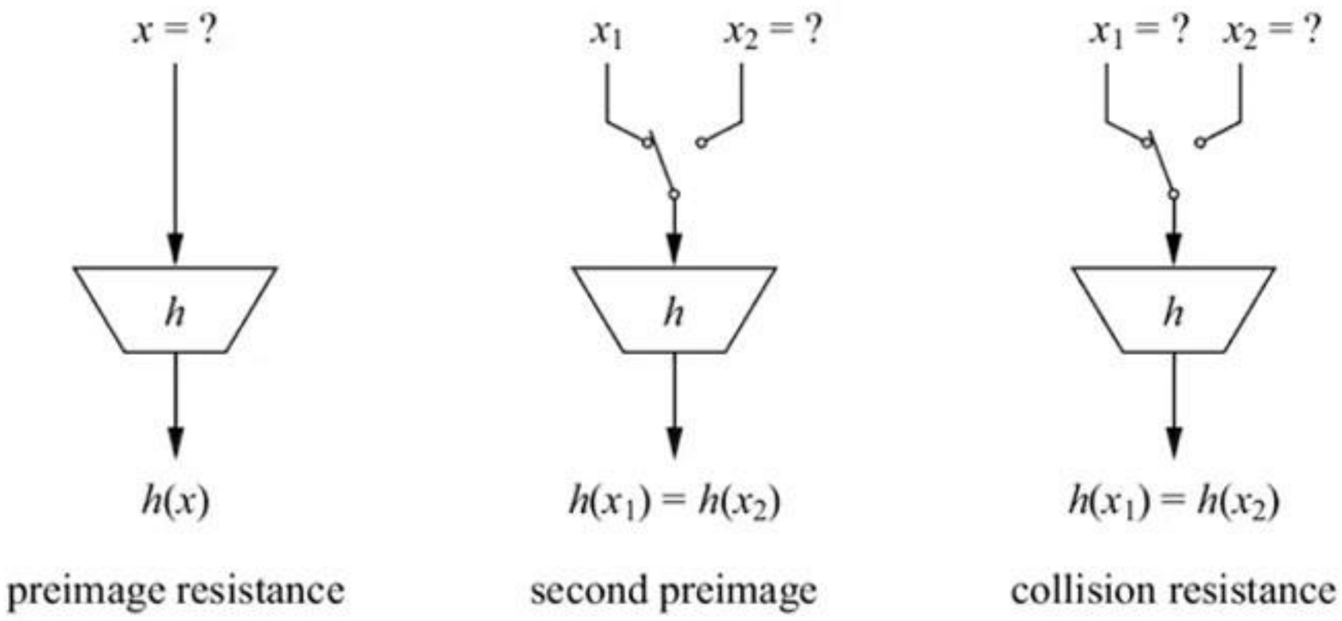
different messages but identical checksums!

Secure Hash Function

- A secure hash function is a mathematical algorithm that takes an input (or 'message') and returns a **fixed-size string of characters**
  - It is designed to be a **one-way function**, meaning it's computationally infeasible to reverse the process (i.e., find the original input from the hash value)

- **Characteristics of Secure Hash Functions**
  1. **Pre-image Resistance**: It should be extremely difficult to find an input that corresponds to a given hash value.
  2. **Collision Resistance**: It should be computationally infeasible to find two different inputs that produce the same hash value.
  3. **Avalanche Effect**: A small change in the input should result in a significantly different hash value.
  4. **Second Pre-image Resistance**: Given a hash value, it should be computationally infeasible to find a different input that produces the same hash value.

Secure Hash Function



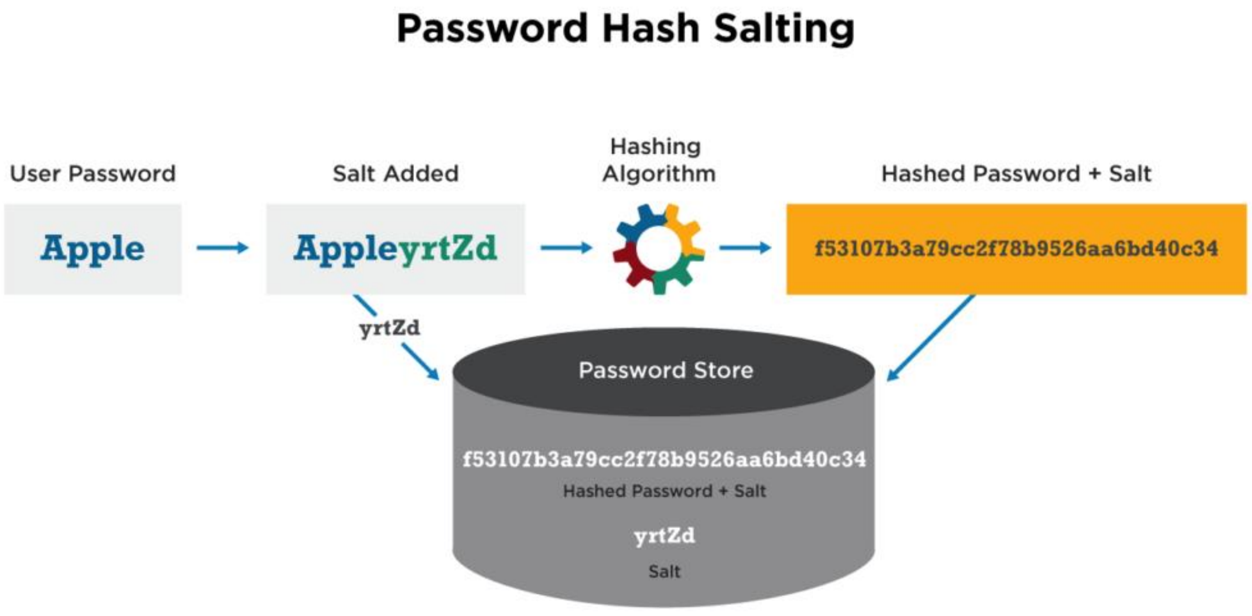
Avalanche Effect

Ivan	EBEC404B6897056F95C343D7E7D2A12E081ADC36413163A03CE3248E9D312C97
ivan	CD0B9452FC376FC4C35A60087B366F70D883FC901524DAF1F122FBD319384F6A

https://academy.moralis.io/blog/what-is-hashing-a-complete-guide-to-hashing

Protecting Passwords Using Hash Functions

- Storing plaintext passwords is insecure; a breach can expose sensitive user information.
- Hashing passwords provides an extra layer of security by transforming them into irreversible, fixed-size values.
  - Add a random value called a "salt" to the password before hashing.
    - Salts make it difficult for attackers to use precomputed hash table.



https://cyberhoot.com/cybrary/password-salting/

Commitment Scheme Using Hash Functions

- A commitment scheme allows one party (the "committer") to commit to a value, keeping it hidden from others, and later reveal the committed value.

- **Components of a Commitment Scheme**

  1. **Commitment Phase**:
     1. The committer selects a value "x" and a random value "r."
     2. Compute a commitment C by hashing both "x" and "r": C = Hash(x || r).
  2. **Reveal Phase**:
     1. At a later time, the committer reveals both "x" and "r."
     2. Others can verify the commitment by hashing "x" and "r" and comparing it to the original commitment.

Commitment Scheme

Commitments {*commit*, *open*, *verify*}

A prover $\mathcal{P}$ hides a secret in the commit phase
and opens it to a verifier $\mathcal{V}$ in the open phase.

Commit phase

$\mathcal{P}$ computes and sends *com to* $\mathcal{V}$:
r = *random*()
com = *commit*(*secret*, *r*)

Hiding

$\mathcal{V}$ cannot find clues of (*secret*, *r*) from *com* at the commit phase.

After some confirmations…

Open phase

$\mathcal{P}$ *open* *secret* and r to $\mathcal{V}$.
$\mathcal{V}$ *verify*(*com*, *secret*, *r*) = *T* / *F*

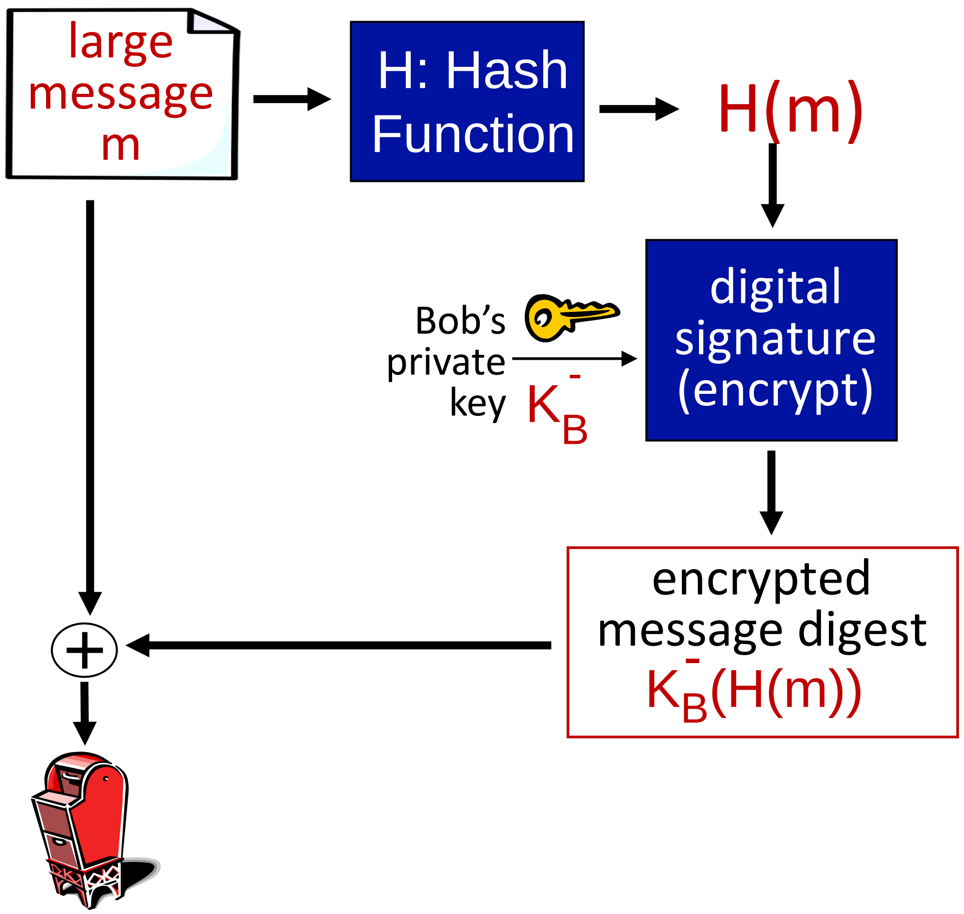
Binding

$\mathcal{P}$ cannnot open (*secret'*, *r'*) =/= (*secret*, *r*) such that
T = *verify*(*com*, *secret'*, *r'*)

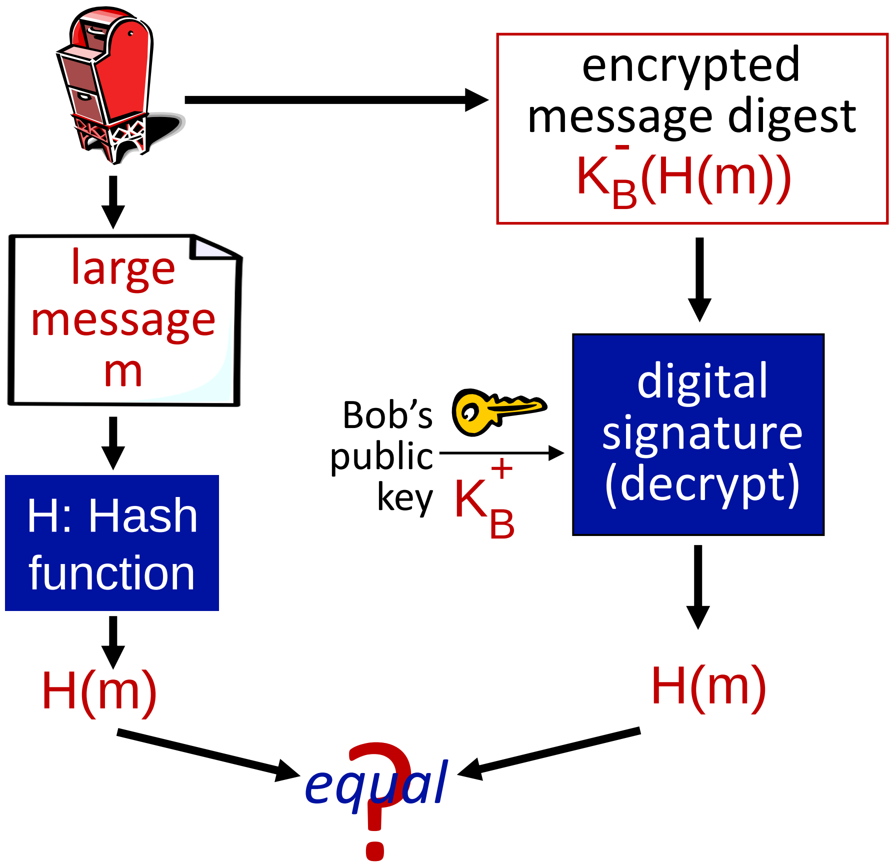
https://kyle-crypto.medium.com/cryptographic-commitments-55ccc4f297c9

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



Types of Hash Functions

- There are two general types of hash functions
  - Dedicated hash functions
    - These are algorithms that are specifically designed to serve as hash functions.
    - MD4, MD5, and SHA family
  - Block cipher-based hash functions
    - It is also possible to use block ciphers such as AES to construct hash functions.
    - Davies–Meyer

Hash function algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest
- SHA2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - outputs 224, 256, 384, and 512 bits, respectively
  - No real security concerns yet

Merkle–Damgard Construction

- The hash value of the input message is then defined as the output of the last iteration of the compression function
- Generally speaking, Let h be a fixed-length hash function for inputs of length $2n$ and with output length n. Construct has function H as follows, then If h is collision resistant, then so is H.

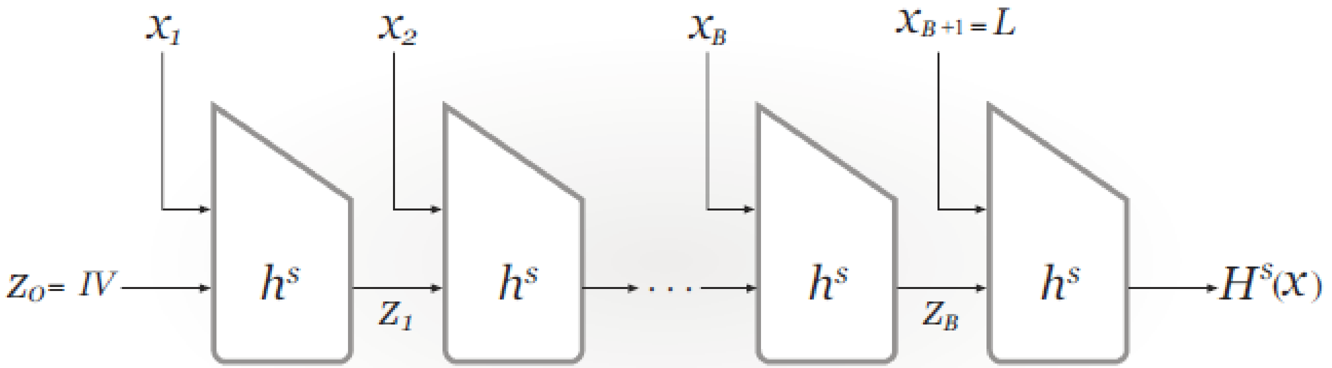


FIGURE 5.1: The Merkle-Damgård transform.

Christof Paar , Jan Pelzl, Understanding Cryptography, 2010.

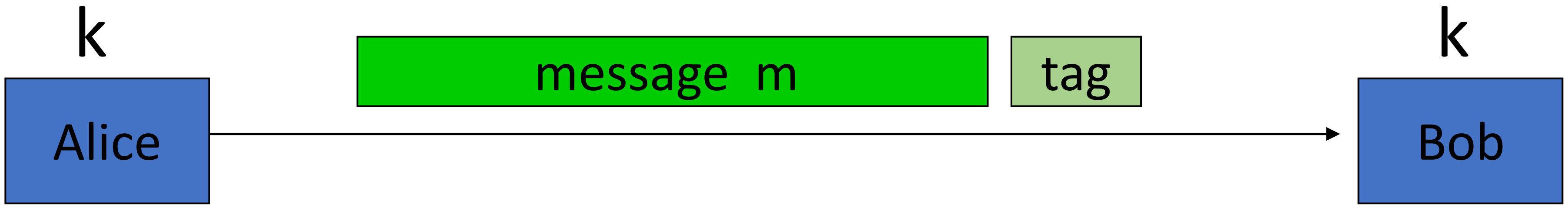
MAC: Message Authentication Code

- Similar to digital signatures, MACs append an authentication tag to a message.
  - The crucial difference between MACs and digital signatures is that MACs use a symmetric key k for both generating the authentication tag and verifying it.
  - A MAC is a function of the symmetric key k and the message x.

$$tag = MAC_k(x)$$

Christof Paar , Jan Pelzl, Understanding Cryptography, 2010.

Message Authentication Codes (MACs)



Generate tag:

$tag \leftarrow S(k, m)$

Verify tag:

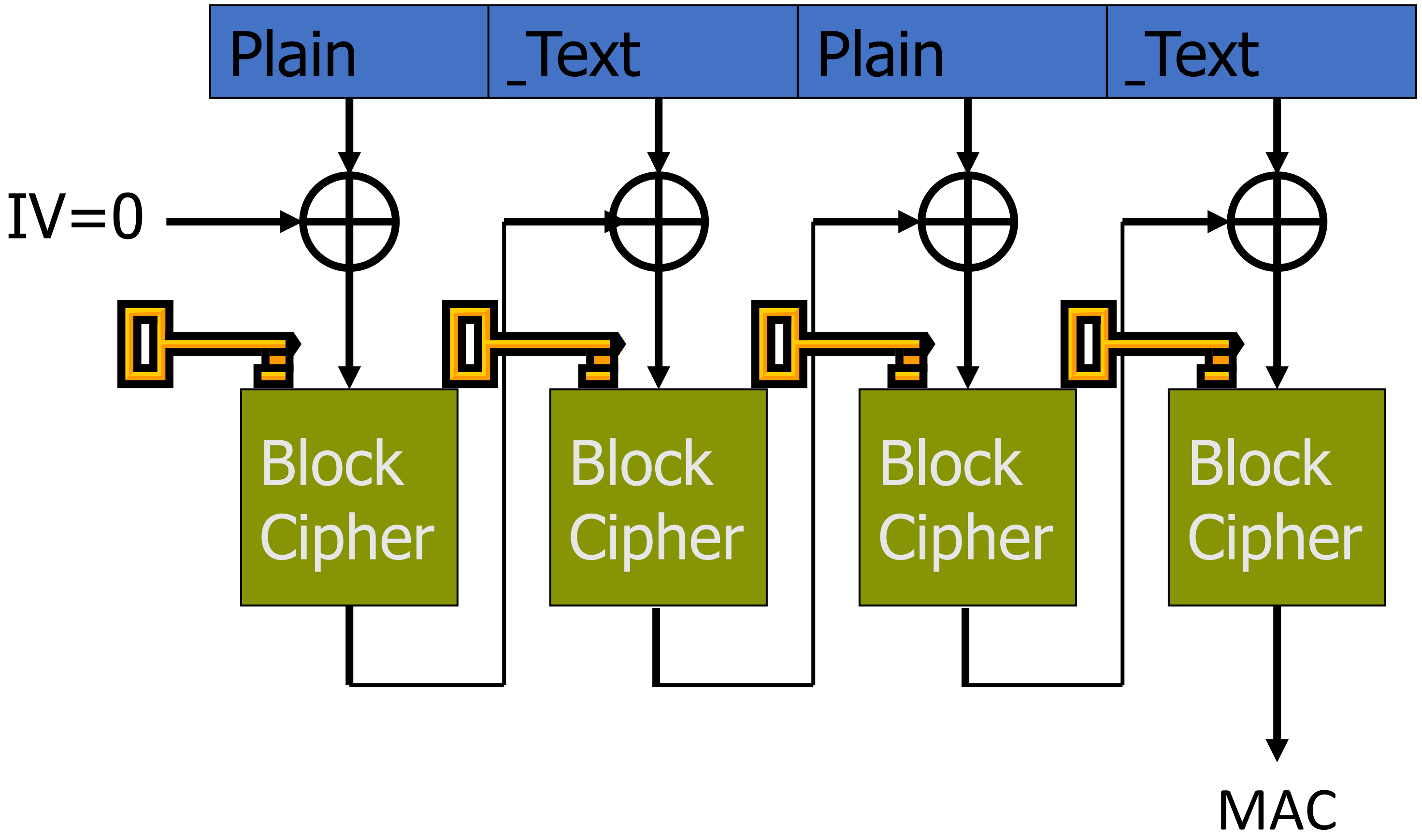
$V(k, m, tag) \stackrel{?}{=}$ `yes'

Def: **MAC** $I = (S,V)$ defined over (K,M,T) is a pair of algs:

- $S(k,m)$ outputs t in T
- $V(k,m,t)$ outputs `yes' or `no'

Basic CBC-MAC

- CBC block cipher, discarding all but last output block
  - It has the same problems as the symmetric key encryption



MACs from Hash Functions: HMAC

- The basic idea behind all hash-based message authentication codes is that the key is hashed together with the message.

$$m = MAC_k(x) = h(k||x)$$

is called secret prefix MAC, and the second one

$$m = MAC_k(x) = h(x||k)$$

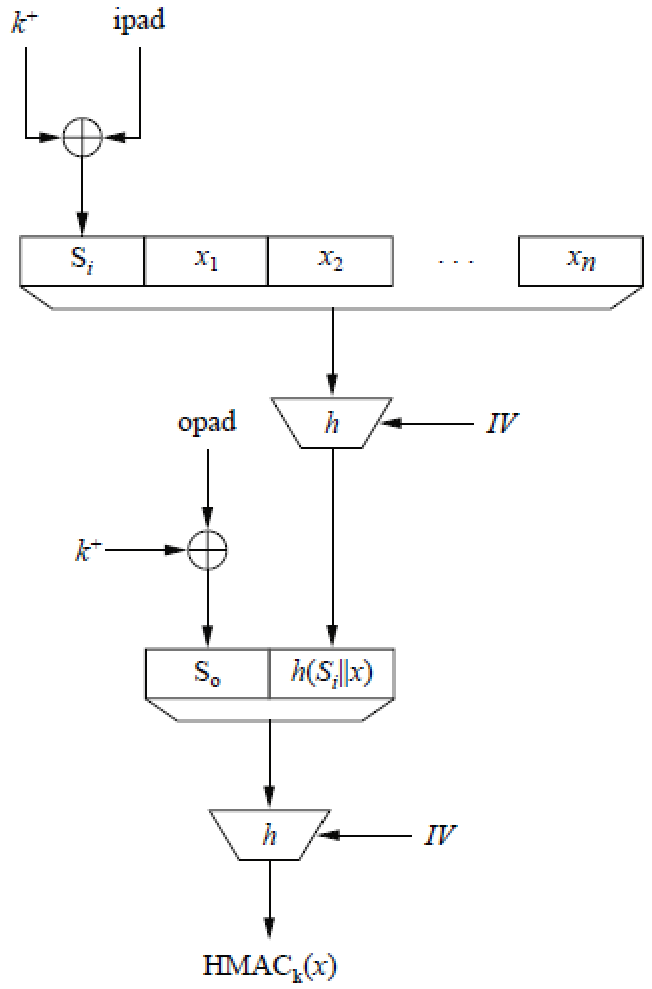
is known as secret suffix MAC.

- Due to the iterative nature of Hash functions, both prefix and suffix MACs have Vulnerabilities.

Christof Paar , Jan Pelzl, Understanding Cryptography, 2010.

HMAC

- A hash-based message authentication code which does not show the security weakness of prefix and suffix MACs construction
  - The scheme consists of an inner and outer hash.



$$HMAC_k(x) = h[(k^+ \oplus opad)||h[(k^+ \oplus ipad)||x]]$$

Christof Paar , Jan Pelzl, Understanding Cryptography, 2010.

MACs vs. Digital signatures

- Key Management
  - Digital Signatures: Involve public and private key pairs.
  - MACs: Require a shared secret key between sender and receiver.
- Verification
  - Digital Signatures: Verifiable by anyone using the sender's public key.
  - MACs: Verification requires possession of the shared secret key.
- Use Case Focus
  - Digital Signatures: Focused on authentication, non-repudiation, and data integrity.
  - MACs: Primarily used for data integrity and authentication.
- MACs have less computational cost and do not need key certification

References

- Kurose, James F., and Keith W. Ross. "Computer networking: A top-down approach edition." Addision Wesley (2007), chapter 8.
- Paar, Christof, and Jan Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.
- ChatGPT, OpenAI.