CSIT 495/595 - Introduction to Cryptography
FINAL Exam - Review

Bharath K. Samanthula
Department of Computer Science
Montclair State University

Topics Covered

- Private-key cryptography
  - Perfect secrecy and one-time pad
  - Block ciphers and their modes of operation
  - MAC and Hash Functions
- Public-key cryptography
  - Number theory overview
  - Key management and exchange protocols
  - RSA
  - ElGamal
  - Digital Signatures

Private-Key Encryption: Formal Definition

- Encryption schemes are defined over the message space $\mathcal{M}$, the key space $\mathcal{K}$, and the ciphertext $\mathcal{C}$

Key Generation (Gen)

- A probabilistic algorithm that selects a key $k \in \mathcal{K}$

Encryption (Enc)

- Input: message $m \in \mathcal{M}$ and k
- Output: Ciphertext $c \leftarrow \mathrm{Enc}_k(m)$, where $c \in \mathcal{C}$

Decryption (Dec)

- Input: c and k
- Output: $m \leftarrow \mathrm{Dec}_k(c)$

Kerckhoff's Principle

- Auguste Kerckhoff, a Dutch Cryptographer, argues the opposite in a paper he wrote in the late 19th century

The cipher method must not be required to be secret, and it must be able to fall into hands of the enemy without inconvenience

- Commonly known as Kerckhoff's Principle
- Encryption should be designed to be secure even if all the details of the encryption scheme are revealed to the adversary, except the key
- In short, security relies solely on the secrecy of the key

Block Ciphers: Modes of Operation

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Output Feedback (OFB)
- Counter (CTR)

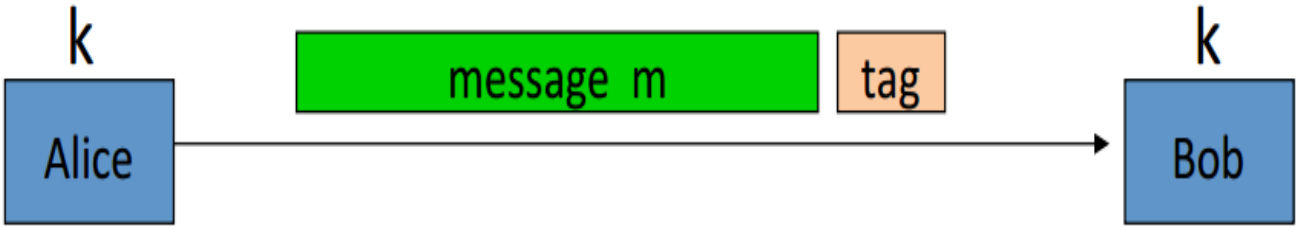
Data Confidentiality vs. Message Integrity

- Do applications always need both confidentiality and integrity? NO
  - protecting OS related files on hard disk
  - protecting banner ads on web pages
- Key observation: encryption schemes that ensure data confidentiality are not necessarily designed to guarantee message integrity
- Never assume that encryption by default solves the problem of message authentication
- Example 1 - Encryption using Stream Ciphers: Suppose $c = k \oplus m$. A single bit flip in c can yield an entirely different message ($\neq m$) upon decryption
- Example 2 - Encryption using Block Ciphers: changing a single bit affects one or more blocks

Message Authentication Code (MAC)

- In general, encryption does not solve the message integrity problem
- *Message Authentication Code (MAC)* enables the receiver to verify *authenticity* of the source and the *integrity* of the received message
- Key question: can we have a single encryption scheme that simultaneously achieves confidentiality and integrity?
  - YES!!.... Authenticated encryption (more details on this later)

MAC: Basic Idea



Generate tag:

$tag \leftarrow S(k, m)$

Verify tag:

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

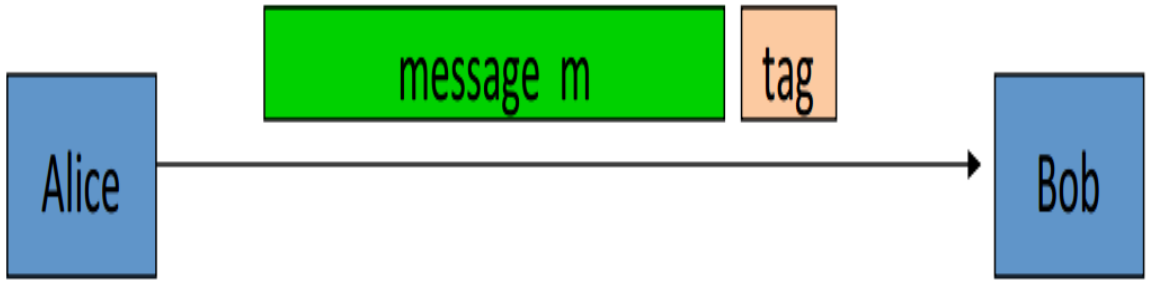
Def: **MAC** I = (S,V) defined over (K,M,T) is a pair of algs:

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

where *S*, *V*: MAC signing and verification algorithms
(K,M,T): key space, message space, and tag space

MAC requires Private Key

Can we use avoid private keys and use error-correcting codes such as CRC to ensure message integrity?



Generate tag:

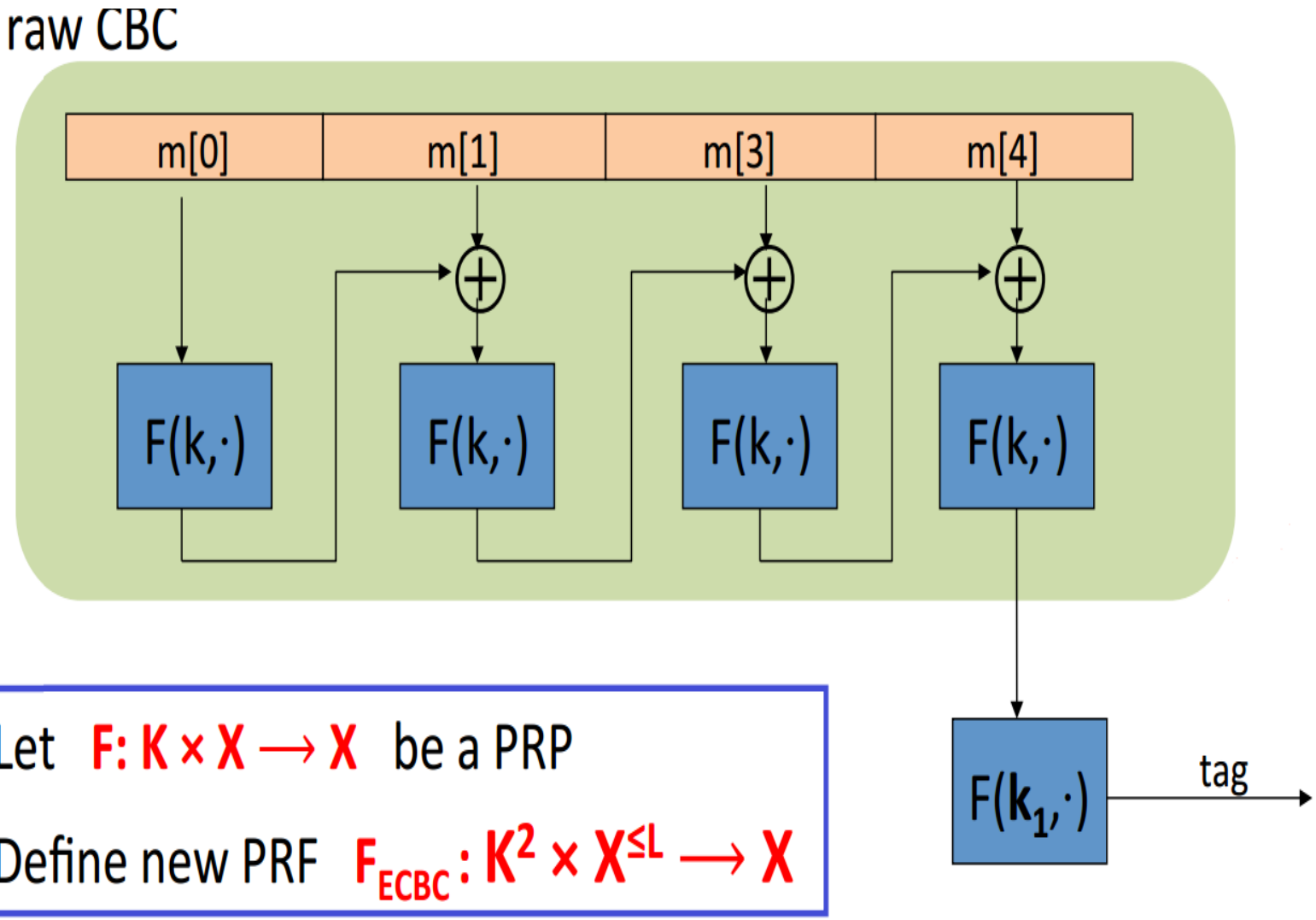
$tag \leftarrow CRC(m)$

Verify tag:

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

- Attacker can easily modify message m and recompute CRC
- For example, attacker can send (m', t') to Bob

Encrypted CBC-MAC (ECBC-MAC)



Let $F: K \times X \rightarrow X$ be a PRP

Define new PRF $F_{ECBC}: K^2 \times X^{\leq L} \rightarrow X$

Encrypted CBC-MAC (ECBC-MAC)

- Key Questions:
  - Is the ECBC-MAC secure without the last encryption?
  - What happens if the message is not a multiple of block size?

Authenticated Encryption Constructions

- Goal: Can we ensure confidentiality and integrity simultaneously by default in the encryption scheme?
- Let k_E and k_M denote encryption and message authentication keys
- Three natural Constructions:
  - Encrypt-and-authenticate
  - Authenticate-then-encrypt
  - Encrypt-then-authenticate

Hash Functions: Motivation

- Hash Function: a function that takes inputs of long length and compress them into short, fixed-length outputs, called *digests* or *hash values*
- Key requirement: avoid collisions for two different inputs that map to the same digest
- Classic Example: hash tables that enable $O(1)$ lookup time

Collision Resistance

Let $H: M \rightarrow T$ be a hash function ($|M| >> |T|$)

A **collision** for H is a pair $m_0, m_1 \in M$ such that:

$$H(m_0) = H(m_1) \quad \text{and} \quad m_0 \neq m_1$$

A function H is **collision resistant** if for all (explicit) "eff" algs. A:

$$\mathbf{Adv_{CR}[A,H] = Pr[\ A \text{ outputs collision for } H]}$$

is "neg".

Example: SHA-256 (outputs 256 bits)

some slides are adopted from *Collision Resistance* by Dan Boneh

MACs from Collision Resistance

$$S^{big}(k, m) = S(k, H(m)) \quad ; \quad V^{big}(k, m, t) = V(k, H(m), t)$$

Collision resistance is necessary for security:

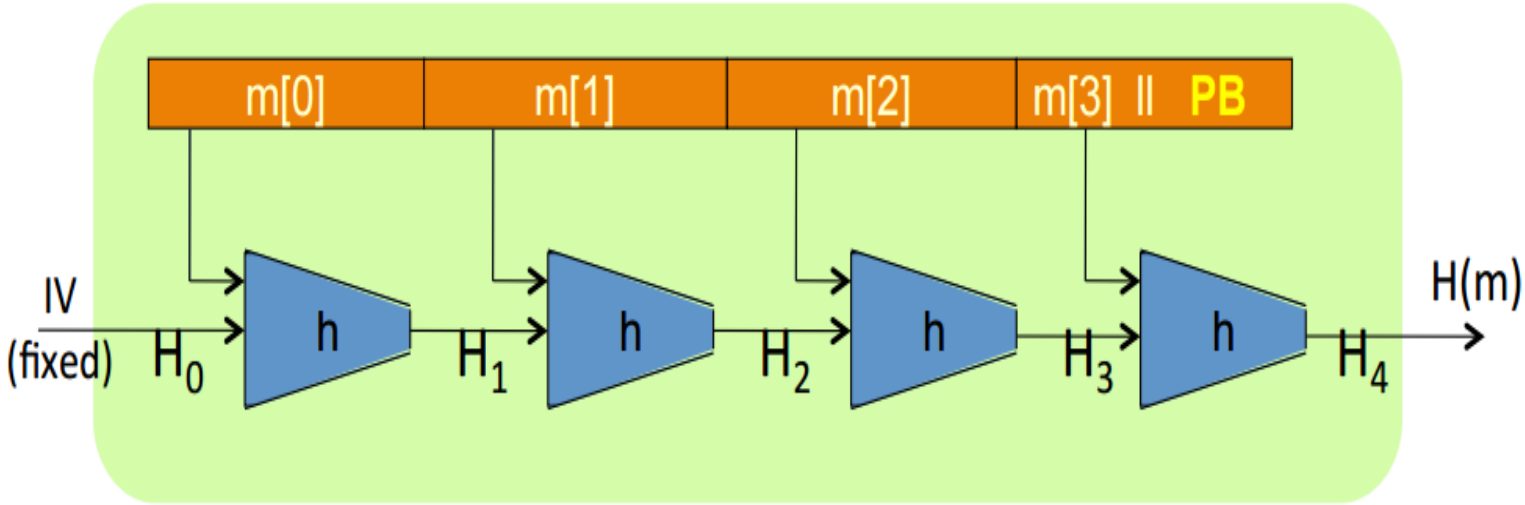
Suppose adversary can find $m_0 \neq m_1$ s.t. $H(m_0) = H(m_1)$.

Then: S^{big} is insecure under a 1-chosen msg attack

step 1: adversary asks for $t \leftarrow S(k, m_0)$

step 2: output (m_1, t) as forgery

The Merkel-Damgard Iterative Construct



Given $h: T \times X \longrightarrow T$ (compression function)

we obtain $H: X^{\leq L} \longrightarrow T$. H_i - chaining variables

PB: padding block

If no space for PB
add another block

Dan Boneh

Standardized Method: Hash-MAC (HMAC)

Most widely used MAC on the Internet.

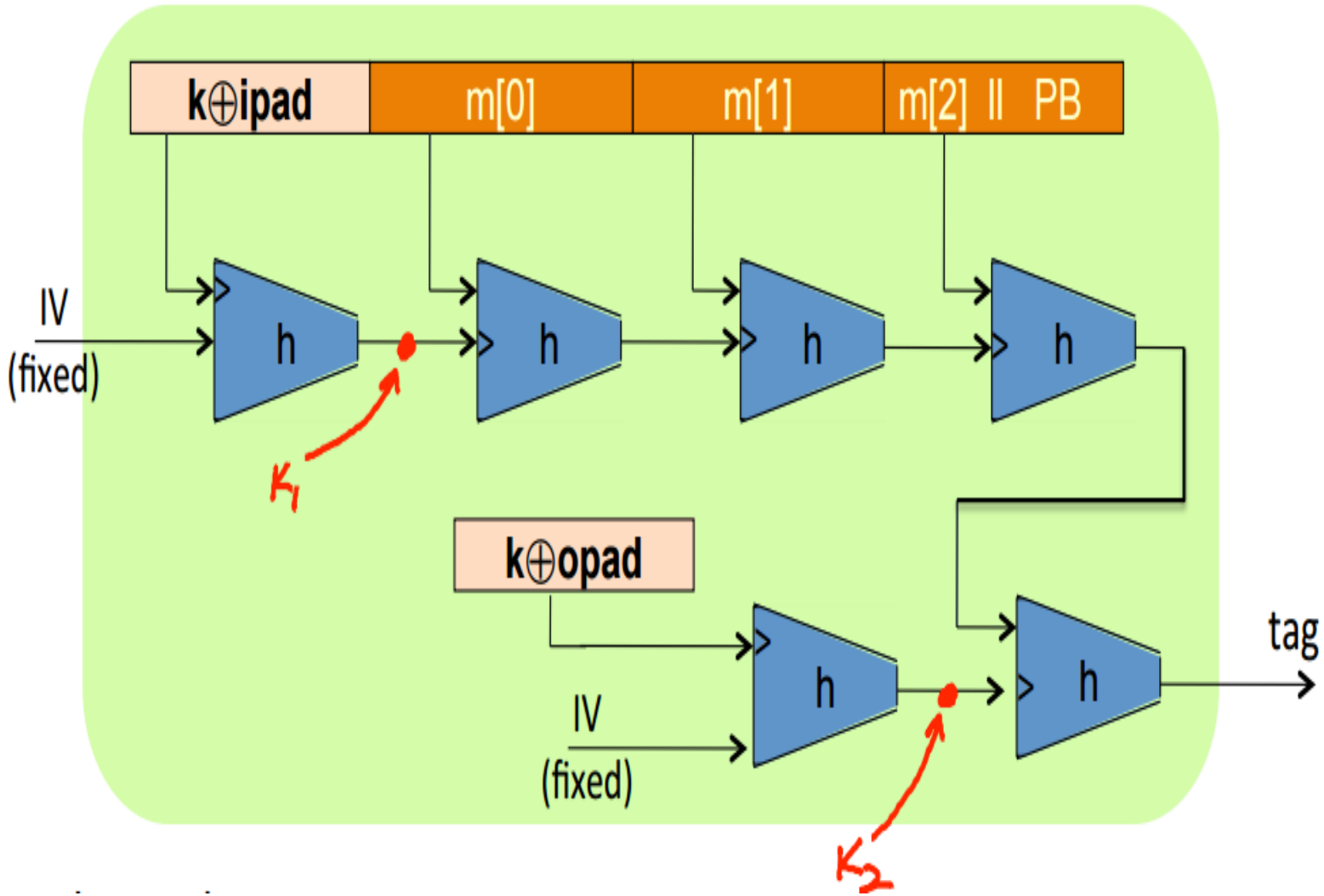
H: hash function.

example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

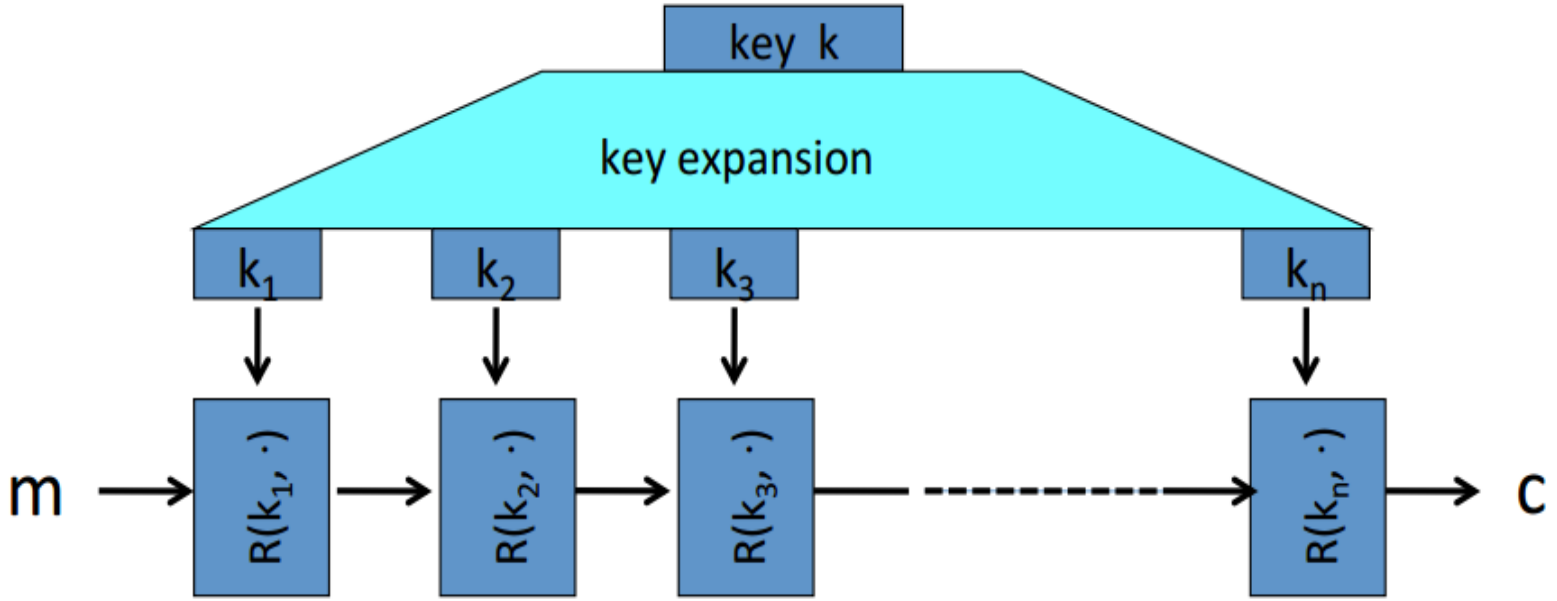
HMAC: $S(k, m) = H(\ k \oplus opad \ \| \ \mathbf{H(\ k \oplus ipad \ \| \ m\ )}\ )$

HMAC: Graphical Interpretation



Block Ciphers: Iterative Approach

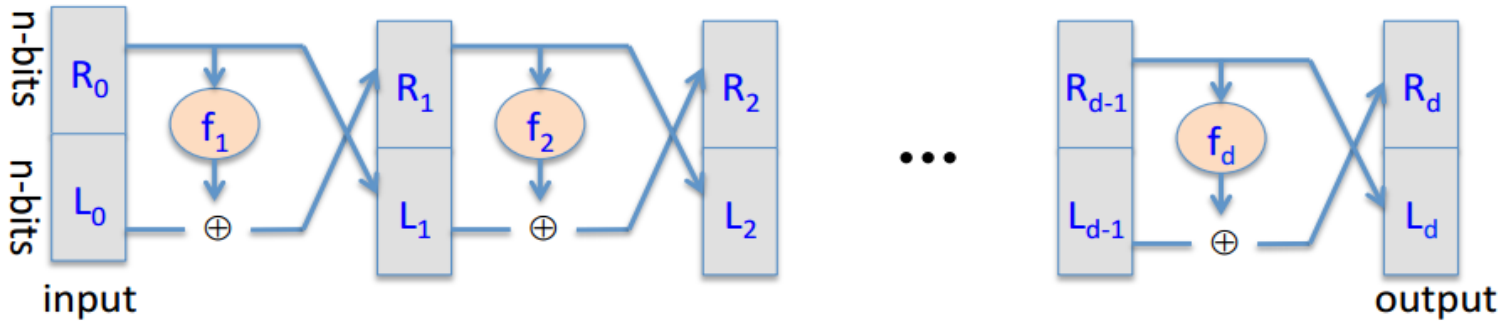
Most modern block ciphers are iterative in nature



R(k,m) is called a round function

for 3DES (n=48), for AES-128 (n=10)

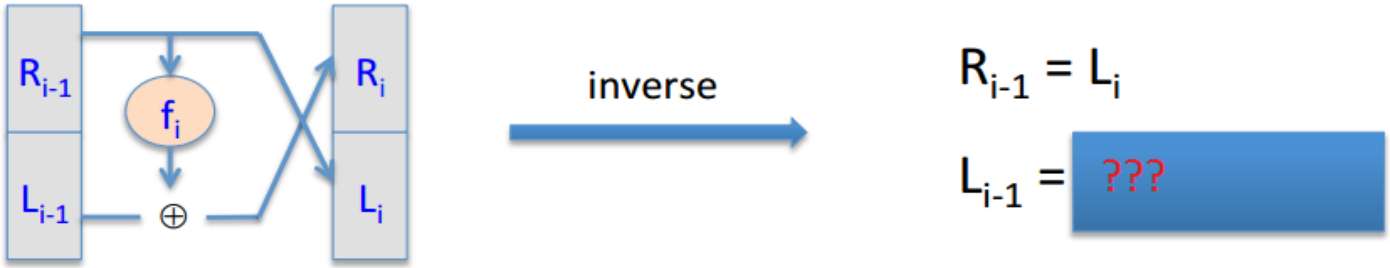
Feistel Network



Claim: for all $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

Feistel network $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is invertible

Proof: construct inverse



Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM
  key-len = 128 bits ; block-len = 128 bits
- 1973: NBS asks for block cipher proposals.
  IBM submits variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
  key-len = 56 bits ; block-len = 64 bits
- 1997: DES broken by exhaustive search
- 2000: NIST adopts Rijndael as AES to replace DES

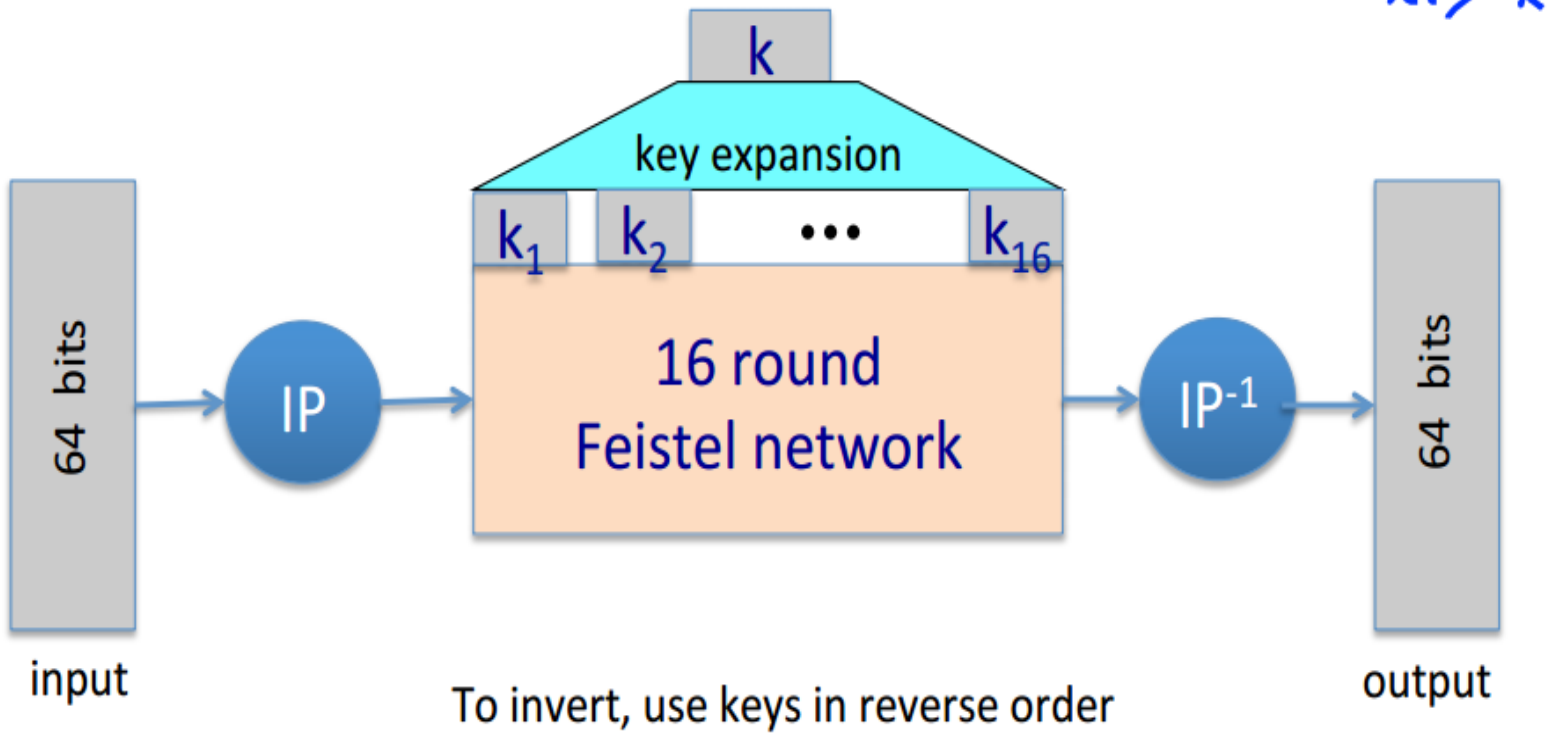
Widely deployed in banking (ACH) and commerce

DES - Graphical Interpretation

Consists of 16 rounds Feistel Network

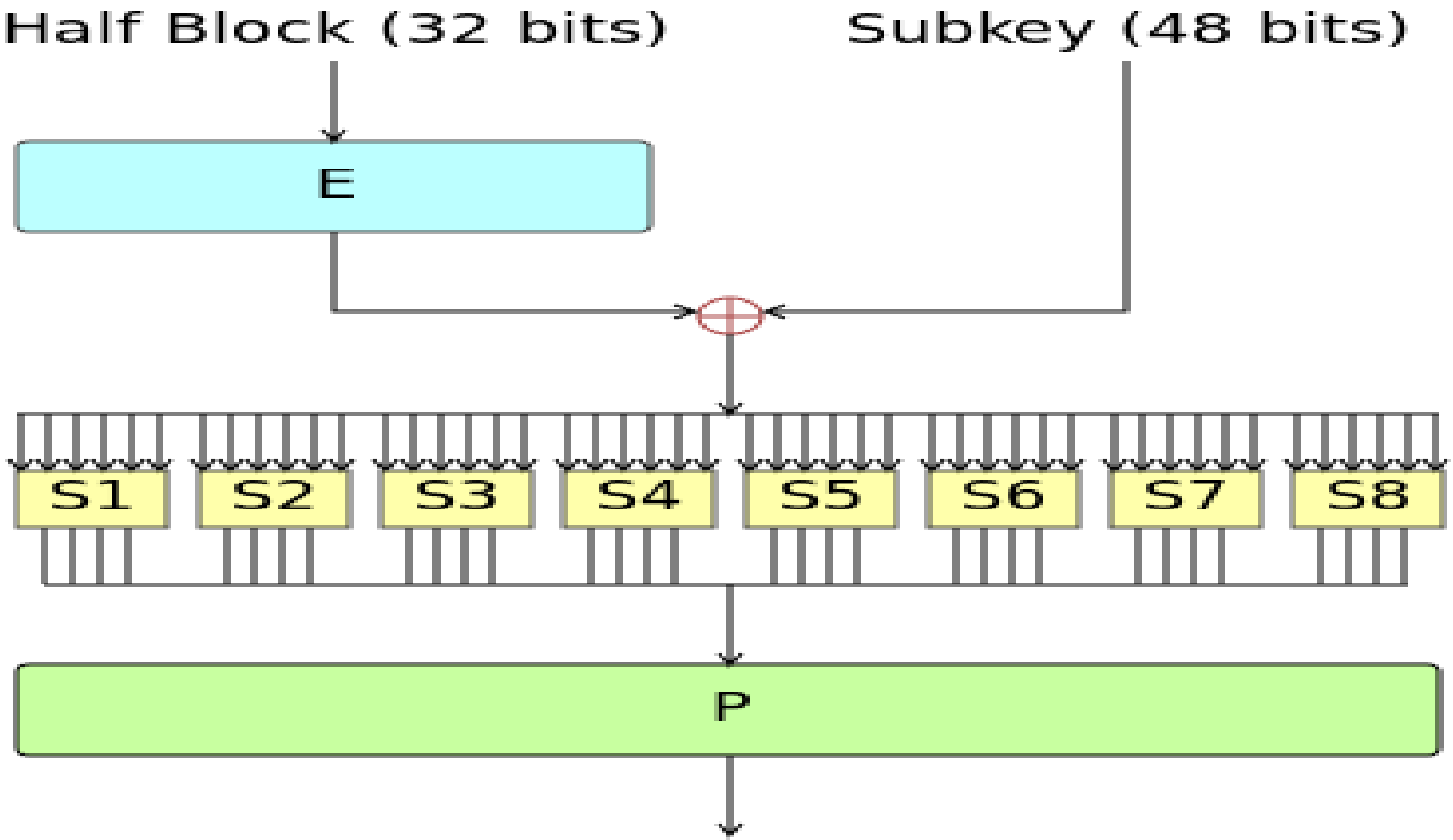
$$f_1, \ldots, f_{16}: \{0,1\}^{32} \longrightarrow \{0,1\}^{32} \quad , \quad f_i(x) = \mathbf{F}(k_i, x)$$

from key K



DES - Graphical Interpretation

$F(k_i, x)$ is constructed as follows



DES - Disadvantages

The best known attack on DES is an exhaustive search through its key space

msg = "The unknown messages is: XXXX … "

CT = c_1 c_2 c_3 c_4

Goal: find $k \in \{0,1\}^{56}$ s.t. $DES(k, m_i) = c_i$ for i=1,2,3

1997: Internet search -- **3 months**

1998: EFF machine (deep crack) -- **3 days** (250K $)

1999: combined search -- **22 hours**

2006: COPACOBANA (120 FPGAs) -- **7 days** (10K $)

⇒ 56-bit ciphers should not be used !! (128-bit key ⇒ 2^{72} days)

Variants of DES

- Modification of Internal Structure – NOT Recommended
- Double DES (2DES) - Double invocation of DES
  - $E'_{k_1,k_2}(m) = E_{k_2}(E_{k_1}(m))$, where E denotes DES encryption
- Key question: Can we use 2DES to improve on the brute-force attacks of DES?
  - NO – It is susceptible to "meet-in-the-middle attack"

Triple-DES (3DES)

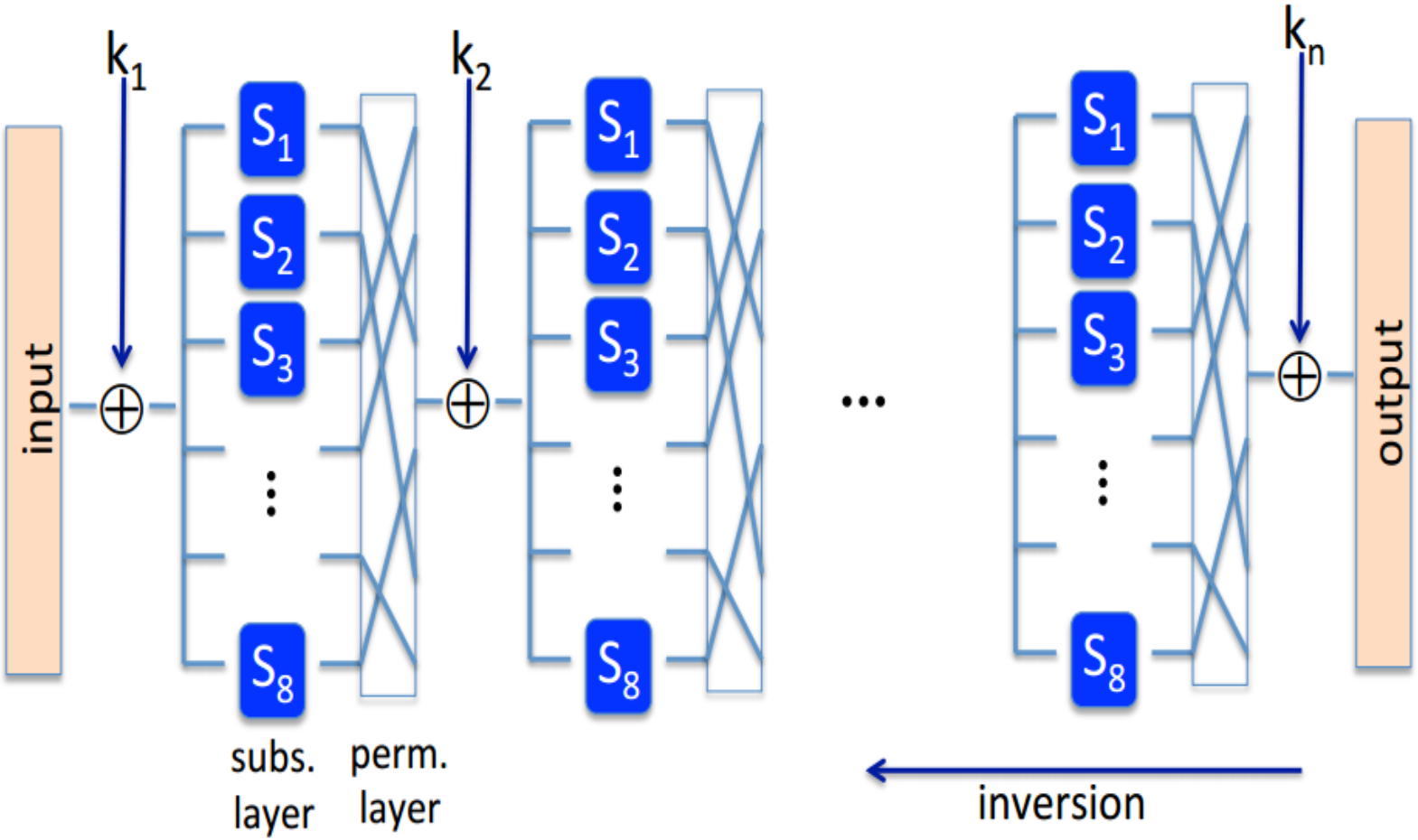
- Triple invocation of DES
  - $E'_{k_1,k_2,k_3}(m) = E_{k_3}(D_{k_2}(E_{k_1}(m)))$
  - Why do we need decryption inside the functionality?
- Standardized in 1999
- Key size: $3x56 = 168$ bits
- 3 times slower than DES

Advanced Encryption Standard (AES)

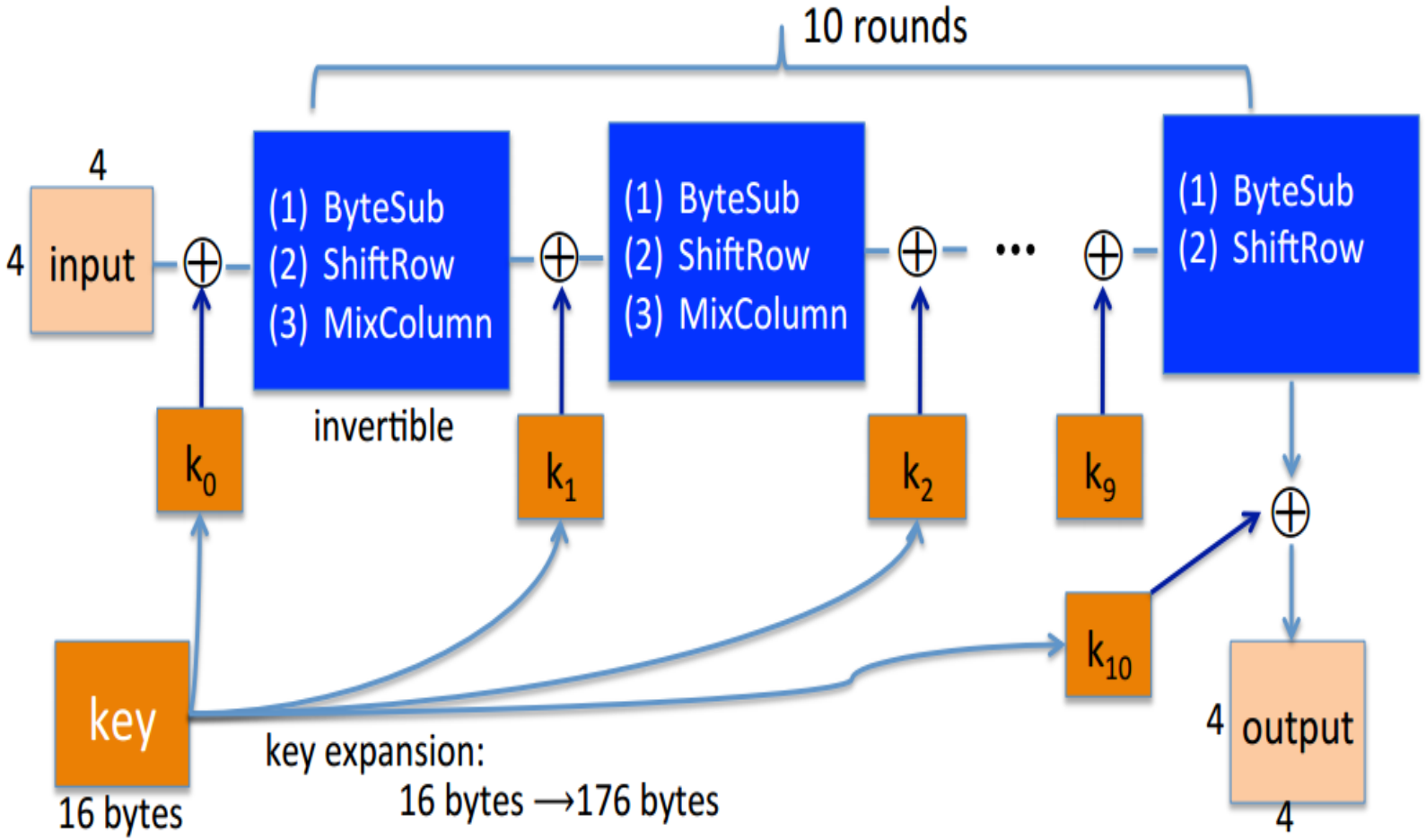
- 1997 - NIST requested for proposals to replace DES
- 1998 - 15 different algorithms were submitted
- 1999 - NIST selected 5 finalists
- October 2000 - NIST announced the winning algorithm, referred to as Rijndael, later standardized as AES
- Key sizes: 128, 192, 256 bits
- Block size: 128 bits

AES

AES is based on substitution permutation network (not Feistel)



AES-128: Basic Steps



AES in Hardware

AES instructions in Intel Westmere:

- **aesenc, aesenclast**: do one round of AES

  128-bit registers: xmm1=state, xmm2=round key

  aesenc xmm1, xmm2 ; puts result in xmm1

- **aeskeygenassist**: performs AES key expansion

- Claim 14 x speed-up over OpenSSL on same hardware

Similar instructions on AMD Bulldozer

MD5

- A hash function with 128-bit output length (proposed in 1991)
- Considered to be collision resistant for some time
- In 2004, a group of Chinese cryptanalysts presented a method to find collisions in MD5
- Nowadays, collisions can be found in MD5 very easily (under one minute on a Desktop PC)
- Although MD5 is still being found in Legacy systems, it should not be used anywhere cryptographic security is needed

SHA Family

- A series of hash functions standardized by NIST
- SHA-0: 160 bits output length, published in 1993, but withdrawn shortly due to serious security flaws
- SHA-1: 160 bits output length, published in 1995, theoretical attack indicates that collisions can be found soon (with fewer than 2^{80} hash function evaluations)
- SHA-2: 256 or 512-bit output length, widely adopted in real-world applications
- October 2012 – NIST announced Keccak or SHA3 as the winner

Topics Covered

- Private-key cryptography
  - Perfect secrecy and one-time pad
  - Block ciphers and their modes of operation
  - MAC and Hash Functions
- Public-key cryptography
  - Number theory overview
  - Key management and exchange protocols
  - RSA
  - ElGamal
  - Digital Signatures

Modular Arithmetic Notation

- Let $\mathbb{Z}_N$ denote the set $\{0,1,\ldots,N-1\}$, where N is a positive integer
- a mod N denote the remainder of a upon division by N
- If $a = qN + r$, then $r = a$ mod N (of course, $q = 0$ if $a \in \mathbb{Z}_N$)
- The process of mapping a to a mod N is called "reduction modulo N"
- Note that a mod $N \in \mathbb{Z}_N$

Modular Inversion

Over the rationals, inverse of 2 is ½. What about $\mathbb{Z}_N$?

Def: The **inverse** of x in $\mathbb{Z}_N$ is an element y in $\mathbb{Z}_N$ s.t. $x \cdot y = 1$ in $\mathbb{Z}_N$

y is denoted x^{-1}.

Example: let N be an odd integer. The inverse of 2 in $\mathbb{Z}_N$ is $\frac{N+1}{2}$

$$2 \cdot \left(\frac{N+1}{2}\right) = N+1 = 1 \text{ in } \mathbb{Z}_N$$

The Group $\mathbb{Z}_N^*$

Def: $\mathbb{Z}_N^*$ = (set of invertible elements in $\mathbb{Z}_N$) =

= { $x \in \mathbb{Z}_N$: gcd(x,N) = 1 }

Examples:

1. for prime p, $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\} = \{1, 2, \ldots, p-1\}$
2. $\mathbb{Z}_{12}^*$ = { 1, 5, 7, 11}

For x in $\mathbb{Z}_N^*$, can find x^{-1} using extended Euclid algorithm.

Fermat's Theorem (1640)

Thm: Let p be a prime

$$\forall\, x \in (Z_p)^* : \quad x^{p-1} = 1 \text{ in } Z_p$$

Example: $p=5$. $\quad 3^4 = 81 = 1$ in Z_5

So: $x \in (Z_p)^* \Rightarrow x \cdot x^{p-2} = 1 \Rightarrow x^{-1} = x^{p-2}$ in Z_p

another way to compute inverses, but less efficient than Euclid

Cyclic Group and Generators

Thm (Euler): $(Z_p)^*$ is a **cyclic group**, that is

$$\exists\ g \in (Z_p)^* \text{ such that } \{1, g, g^2, g^3, \ldots, g^{p-2}\} = (Z_p)^*$$

g is called a **generator** of $(Z_p)^*$

Example: p=7. $\{1, 3, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\} = (Z_7)^*$

Not every elem. is a generator: $\{1, 2, 2^2, 2^3, 2^4, 2^5\} = \{1, 2, 4\}$

Group Order

For $g \in (Z_p)^*$ the set $\{1, g, g^2, g^3, \ldots\}$ is called

the **group generated by g**, denoted $\langle g \rangle$

Def: the **order** of $g \in (Z_p)^*$ is the size of $\langle g \rangle$

$$\mathbf{ord_p(g) = |\langle g \rangle| = (\text{smallest } a>0 \text{ s.t. } g^a = 1 \text{ in } Z_p)}$$

Examples: $\text{ord}_7(3) = 6$; $\text{ord}_7(2) = 3$; $\text{ord}_7(1) = 1$

Thm (Lagrange): $\forall g \in (Z_p)^*$: $\mathbf{ord_p(g)}$ divides $p-1$

Euler's Generalization of Fermat

Def: For an integer N define $\varphi(N) = \left|(Z_N)^*\right|$ (Euler's φ func.)

Examples: $\varphi(12) = \left|\{1,5,7,11\}\right| = 4$; $\varphi(p) = p-1$

For $N=p \cdot q$: $\varphi(N) = N-p-q+1 = (p-1)(q-1)$

Thm (Euler): $\forall\, x \in (Z_N)^*$: $x^{\varphi(N)} = 1$ in Z_N

Example: $5^{\varphi(12)} = 5^4 = 625 = 1$ in Z_{12}

Generalization of Fermat. Basis of the RSA cryptosystem

Intractable problems

- There many problems that are easy to solve
  - Given a composite N and $x \in \mathbb{Z}_N$ find x_N^{-1}
- There also exist several hard (intractable) problems
  - Discrete Logarithm (DLOG)
  - Factorization

Private-Key vs. Public-Key

Private-Key	Public-Key
Symmetric	Asymmetric
All keys remain private	Only secret key remains private
Secret key is used for communication between those two parties	Multiple senders can communicate with a receiver using his public key

- Observation 1: Public-key encryption is roughly 2 to 3 orders of magnitude slower than private-key encryption
- Observation 2: It can be a challenge to implement public-key encryption in resource-constrained devices, such as smartcards and wireless sensors

Public-Key Encryption: Definition

Def: a public-key encryption system is a triple of algs. (G, E, D)

- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$ (pk, sk) output by G :

$$\forall m \in M: \quad D(sk, E(pk, m)) = m$$

RSA Key Generation

- Selecte two large prime numbers of the same size p, q
- Compute $N = p * q$ and $\Phi(n) = (p-1)(q-1)$
- Select a random integer e such that $1 < e < \Phi(n)$ and $\gcd(e, \Phi(n)) = 1$
- Compute d such that $1 < d < \Phi(n)$ and $ed = 1 \bmod \Phi(n)$
- Output:
  - Public key $pk = (N, e)$
  - Private Key $sk = d$

Textbook RSA Encryption Scheme

- Encryption:

  given a message m, where $0 < m < N$
  use $pk = (N, e)$ to encrypt it
  $c = m^e \bmod N$

- Decryption:

  given a ciphertext c, use $sk = d$ to decrypt it
  $m = c^d \bmod N = m^{ed} \bmod N = 1$,

Textbook RSA: Example

- Let $p = 17$ and $q = 23 \rightarrow N = 391$
- $\Phi(391) = 16 * 22 = 352$
- Suppose $e = 3$, then $d = 235$
- Public key: (391, 3) and the private key is 235
- Given $m = 158$:
  - *Encryption*: $c = (158)^3 \bmod 391 = 295$
  - *Decryption*: $c^d \bmod 391 = (295)^{235} \bmod 391 = 158$

Is Textbook RSA Secure??

- Never use textbook RSA
- Key Observation: Textbook RSA is deterministic, that is, it is not CPA-secure
- What is the Solution?
  - Padded-RSA: choose a uniform bit-string r and encrypt $m' = r||m$ instead of m

Key Distribution and Management

- In private-key cryptography, parties communicate using a shared secret key
- Key Question: How can the parties share a secret key in the first place?
- Naive approaches:
  1. two parties can meet and exchange the key
  2. parties use a trusted courier service as secure channel
- The above approaches were used earlier to share keys in many government, diplomatic and military contexts (e.g., the "red phone" connecting Moscow and Washington in 1960s)

How Convenient is this Secure Channel?

Example: Consider a large multinational company X

- Suppose every pair of employees at X want to securely communicate
- It is difficult for each pair of employees to meet to share the key (often, this may become impossible for employees working in different continents)
- Also, how to share keys with new employees?

Key Management Issues

- If N employees are able to share secret keys, then each employee will have to manage and store $N - 1$ keys
- There may be other keys the employee has to manage (for connecting to printers, customers, etc.)
- The more keys there are, the harder it is to protect them
- Of course, all the keys should be stored securely
- In case of fewer keys, keys can be stored on *secure hardware* such as a *smartcard*

Issues with Private-Key Cryptography

Broadly, three main issues:

- key distribution
- storing and managing large number of secret keys
- inapplicability of private-key cryptography to open systems

In open systems, users may not be aware of each others' existence until they want to communicate (e.g., sending an email to someone whom you have never met)

Key Distribution Techniques

- Key-Distribution Center (KDC)
- Diffie-Hellman Key Exchange

Key-Distribution Center (KDC) 1

- Consider the previous example where all pairs of employees must be able to communicate securely
- Assume there is a key distribution center (a trusted party such as system administrator) who can establish shared keys
- A naive approach
  1. For any new employee i, the KDC can share a key with that employee on his first day of work (at a secret location)
  2. KDC generates $i - 1$ keys, say $k_1, \ldots, k_{i-1}$, and sends k_j to j^{th} employee
- Is the above approach practical?

Key-Distribution Center (KDC) 2

A better Approach

- KDC generates keys on an on-demand basis
- Suppose KDC shares k_A and k_B with Alice and Bob, resp.
- If Alice wants to send a message to Bob, she informs the KDC
- KDC generates a session key and sends this to Alice encrypted using k_A and Bob using k_B
- After finishing the conversation, Alice and Bob can erase the session key

Key-Distribution Center (KDC) 3

Pros

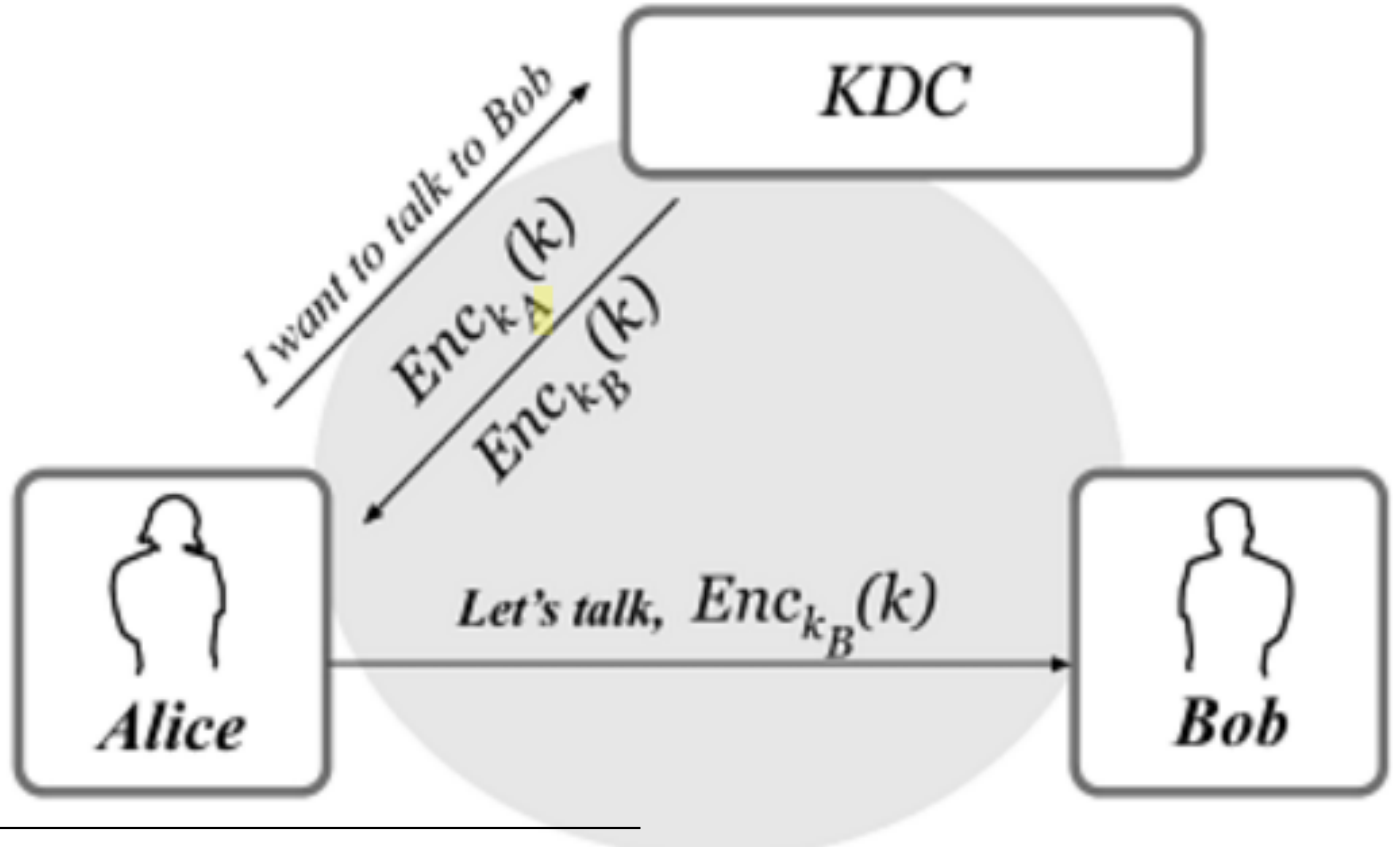
- Each employee need to store only one long-term key (the one they share with KDC)
- If a new employee joins the company, the work is minimal - a key must be set up between this employee and KDC

Cons

- A successful attack on KDC can compromise the whole system
- KDC acts as a single point of failure (if KDC is down due to over load, secure communication may be temporarily impossible)

Needham-Schroeder Protocol 1

- Forms the core of Kerbos - a widely used service[1] in many universities for authentication and supporting secure communication



[1] Kerberos is the default mechanism for supporting secure networked communication in Windows and many UNIX systems.

Needham-Schroeder Protocol 2

- The second ciphertext, $Enc_{k_B}(k)$, is sometimes called a ticket
- In this protocol, KDC need not initiate a second connection to Bob
- Also, if Bob is offline, KDC need not to worry
- Once Alice retains the ticket, she can re-initiate a secure communication with Bob without the involvement of Bob

Diffie-Hellman Protocol 1

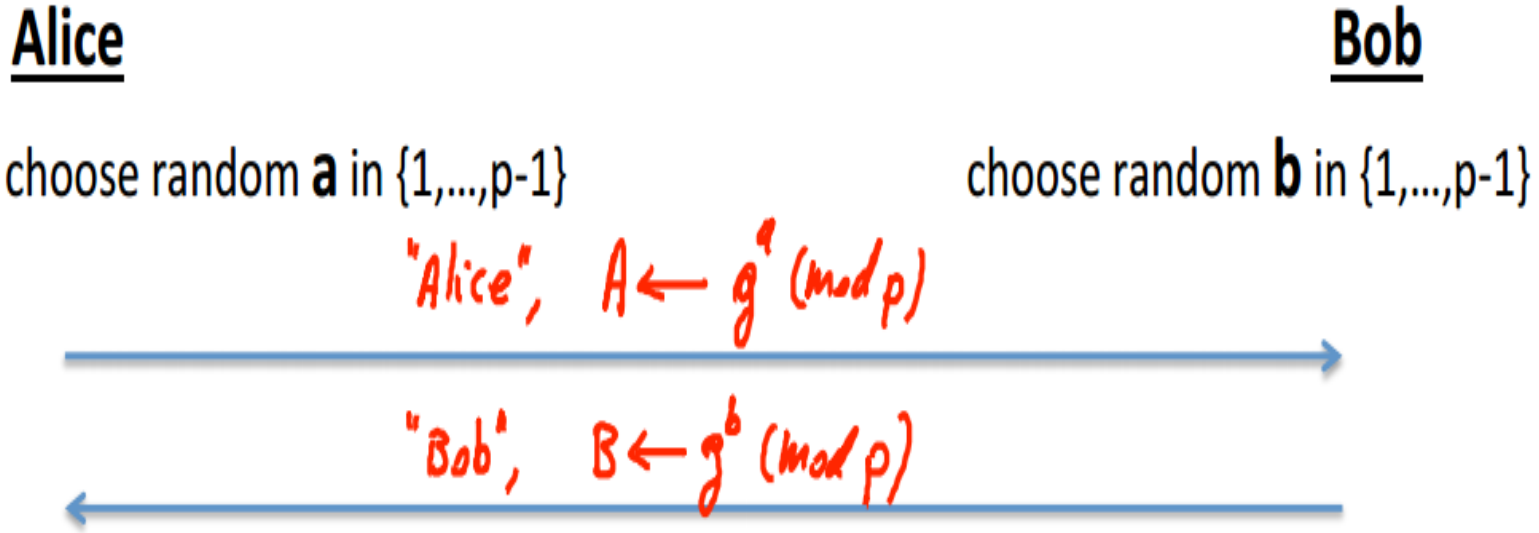
- KDCs and protocols like Kerberos still require, a private and authenticated channel at some point
- Key Question: Can we achieve private communication without ever communicating over a private channel?
- In 1976, *Whitfield Diffie* and *Martin Hellman* published a paper titled “New Directions in Cryptography”
  - Observed that certain things can be easily performed but not easily reversed
  - Example: it is easy to multiply two large prime numbers, but extremely hard to recover their primes from product (Factoring problem)

Diffie-Hellman Protocol 2

Based on the idea that parties can perform operations that they can reverse but that an eavesdropper cannot

Fix a large prime p (e.g. 600 digits)

Fix an integer g in {1, ..., p}



$$B^a \pmod p = (g^b)^a = k_{AB} = g^{ab} \pmod p = (g^a)^b = A^b \pmod p$$

Diffie-Hellman Protocol: Example

1. Alice and Bob agree on $p = 23$ and $g = 5$.
2. Alice chooses $a = 6$ and sends $5^6 \mod 23 = 8$.
3. Bob chooses $b = 15$ and sends $5^{15} \mod 23 = 19$.
4. Alice computes $19^6 \mod 23 = 2$.
5. Bob computes $8^{15} \mod 23 = 2$.

Then 2 is the shared secret.

Diffie-Hellman Protocol: Security

Eavesdropper sees: $p, g, A=g^a \pmod p$, and $B=g^b \pmod p$

Can she compute $g^{ab} \pmod p$??

The above DH problem (involve solving discrete logarithm) is considered to be hard for groups whose order is large enough

ElGamal Encryption: Introduction

- Introduced by *Taher El Gamal* based on the Diffie-Hellman key-exchange protocol (1985)
- Security is based on discrete logarithm and Decisional Diffie-Hellman (DDH) assumptions
- The ciphertext size is twice that of original message (Note: this is not the case in RSA)
- Uses different randomization in each encryption - each message has many different possible ciphertexts

ElGamal: Key Generation

- Suppose Alice wants to generate public and private keys based on ElGamal
- Public key: (p, g, A), where $\mathbb{Z}_p$ represents a cyclic group of order q and g is the generator
  - $A = g^a \bmod p$ and a is randomly chosen in $\mathbb{Z}_q$ by Alice
- Private Key: a

ElGamal: Encryption + Decryption

- Suppose Bob wants to send a message m to Alice
- Note that Bob knows Alice public key (p, g, A)
- Encryption by Bob:
  - Choose a uniform $b \in \mathbb{Z}_q$
  - Compute $c_1 = g^b \bmod p$ and $c_2 = A^b \cdot m \bmod p$
  - Send ciphertext $\langle c_1, c_2 \rangle$ to Alice
- Decryption by Alice:
  - Using private key a, compute $\hat{m} = \frac{c_2}{c_1^a}$

Why ElGamal Encryption Scheme Works

Expand decryption process:

- $$\begin{aligned}\hat{m} &= \frac{c_2}{c_1^a} \\ &= \frac{A^b \cdot m}{(g^b)^a} \\ &= \frac{(g^a)^b \cdot m}{(g^b)^a} \\ &= \frac{g^{ab} \cdot m}{g^{ab}} \\ &= m\end{aligned}$$

ElGamal Encyrption Scheme: Example (1)

- Let Alice choose G with prime $p = 107$, $g = 2$ and $a = 67$
- Alice compute $A = g^a = 2^{67} \bmod 107 = 94$
- Alice Public key: $(107, 2, 94)$
- Alice private key: 67

ElGamal Encyrption Scheme: Example (2)

- Suppose Bob wants to send message “66” to Alice
- Say Bob chooses a random integer $b = 45$
- Encryption by Bob:
  $c_1 = g^b \bmod 107 = 2^{45} \bmod 107 = 28$
  $c_2 = A^b \bmod 107 = 94^{45} \bmod 107 = 9$
- Decryption by Alice:
  Compute $(c_1^a)^{-1} \cdot c_2 \bmod 107 = (28^{67})^{-1} \cdot 9 \bmod 107 = 66$

Analysis of ElGamal (1)

- a (chosen at random) must be kept secret by Alice
- b is a random integer:
  - $c_1 = g^b \bmod p$ remains a random integer
  - A^b is also a random integer mod p
  - Therefore, $c_2 = A^b \cdot m \bmod p$ is the message m multiplied by a random integer
- What happens if b is know to the attacker?

Analysis of ElGamal (2)

Sender must use different b values while encrypting each message (even when encrypting the same message at different times)

- Suppose Bob uses same b for encrypting two messages m_1 and m_2
- In this case, Bob sends $\langle g^b, A^b \cdot m_1 \rangle$ and $\langle g^b, A^b \cdot m_2 \rangle$ for m_1 and m_2, resp.
- This reveals lot to information to the attacker listening on the communication channel. For example,
  - $\frac{m_1}{m_2}$ is known to the attacker
  - Further, if the attacker finds out m_1, he can also determine m_2

Overhead of ElGamal

- Encryption: Two exponentiations; preprocessing possible
- Decryption: one exponentiation
- Message expansion: ciphertext is twice the length of plaintext

Private-Key vs. Public-Key: Recap

	Private-Key Setting	Public-Key Setting
Secrecy	Private-key encryption	Public-key encryption
Integrity	MACs	Digital Signatures

- In public-key setting, Integrity/authenticity is achieved using digital signatures
- Digital Signatures can be considered as the public-key analogue to MACs (with some differences)

Digital Signatures: Basic Idea

- Consider a user Alice (often called *signer* in this context) owns a public-private key pair
- Suppose Alice wants to send a message m to Bob
- Assume that Bob knows the public key of Alice (this can be achieved using techniques discussed earlier)
- Key Goal of Signature Schemes:
  1. Alice signs the message and sends it to Bob
  2. Bob should verify that m was originated from Alice and was not modified in transit
- In general, anyone with Alice's public key should be able to verify m
- Observation: In the context of digital signatures, the owner of the public-key is the sender (note that this is not the case in a typical public-key encryption)

MACs vs. Digital Signatures (DS in short)

- MAC: sender need to compute a separate MAC for each receiver
- DS: compute only a single signature for each message sent to multiple receivers

- MAC: only the intended receiver with the corresponding secret key can validate the message
- DS: Publicly Verifiable (implies that signatures are transferable)

MACs vs. Digital Signatures (DS)

- MAC: cannot provide non-repudiation
- DS: provide non-repudiation

- Key Observation 1: MACs are shorter and roughly 2-3 orders of magnitude faster than digital signatures
- Key Observation 2: If the sender is sending a message only to a single party and if public verifiability, transferability, and/or non-repudiation is required, **MAC should be used**

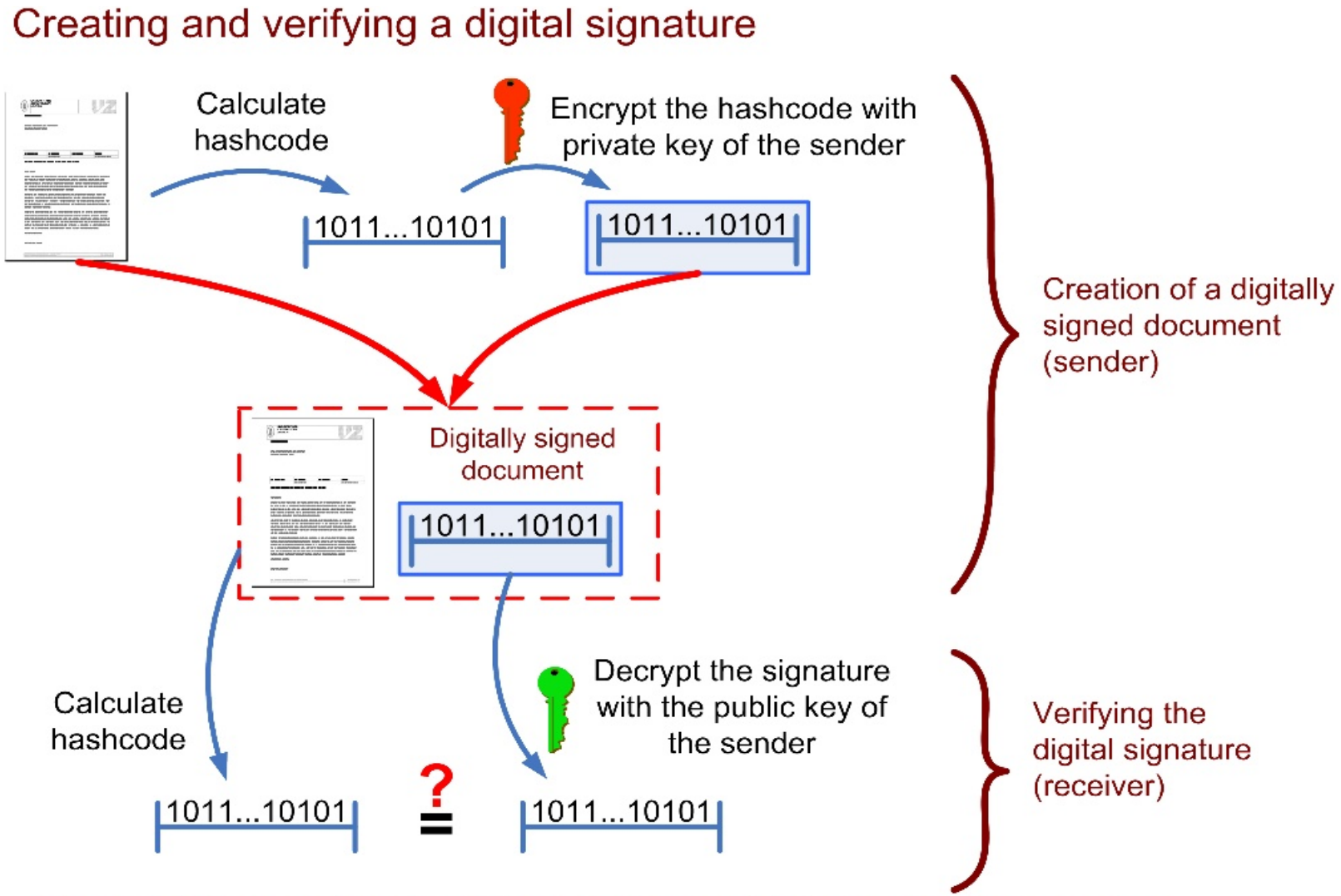
Digital Signatures: Definition

Consists of three algorithms

- Key Generation: takes a security parameter (usually key length) and outputs public key *pk* and private key *sk*
- Signing: takes a message *m* and private key *sk* and generates a signature *S*
- Verification: takes public key *pk*, message *m* and a signature *S* as inputs and outputs 1 if valid, and 0 otherwise

Security: the adversary should not be able to generate a forgery with a probability better than a negligible function

Hash-and-Sign Approach



RSA-Based Digital Signatures (1)

Key generation (as in RSA encryption):

- Select 2 large prime numbers of about the same size, p and q
- Compute $n = pq$, and $\Phi = (q - 1)(p - 1)$
- Select a random integer e, $1 < e < \Phi$, s.t. $\gcd(e, \Phi) = 1$
- Compute d, $1 < d < \Phi$ s.t. $ed \equiv 1 \bmod \Phi$

Public key: (e, n)

Secret key: d, p and q must also remain secret

RSA-Based Digital Signatures (2)

Signing message M

- M must verify $0 < M < n$
- Use private key (d)
- compute $S = M^d \bmod n$

Verifying signature S

- Use public key (e, n)
- Compute $S^e \bmod n = (M^d \bmod n)^e \bmod n = M$

Note: in practice, a hash of the message is signed and not the message itself.

RSA-Based Digital Signatures: Security

Example of forging

- Attack based on the multiplicative property of property of RSA.

  $y_1 = sig_K(x_1)$

  $y_2 = sig_K(x_2)$, then

  $ver_K(x_1x_2 \bmod n, y_1y_2 \bmod n) = true$

- So adversary can create the valid signature

  $y_1y_2 \bmod n$ on the message $x_1x_2 \bmod n$
- This is an existential forgery using a known message attack.

Methods based on DLP: ElGamal and Schnorr signature schemes

Digital Certificates

- Important application of digital signatures
- Use public-key cryptography to securely distribute public keys
- A single public-key of a trusted-party, called *certificate authority*, is first distributed in a secure fashion. Then this key can be used to distribute many other public keys
- Digital certificate: signature binding an entity to some public key

Digital Certificates: Basic Idea

- Consider two users Charlie and Alice with (pk_C, sk_C) and (pk_A, sk_A)
- Assume that Charlie knows Alice's public key pk_A
- Charlie Computes:

$$Cert_{C \to A} = Sign_{sk_C}(Alice's\ public\ key\ is\ pk_A)$$

We call $Cert_{C \to A}$ a certificate for Alice's pubic key pk_A issued by Charlie (trusted or certificate authority)

- In general, other details are used in the certificate generation, such as Alice's
  - full name
  - email address
  - URL of personal website

Digital Certificates: Basic Idea

- Suppose Bob wants to communicate with Alice
- Assume Bob knows pk_C
- Alice sends $(pk_A, Cert_{C \to A})$ to Bob
- Bob verifies whether $Cert_{C \to A}$ is a valid signature on the message *Alice's public key is* pk_A with respect to pk_C
- If the verification succeeds, Alice and Bob can communicate over an insecure and authenticated channel

Digital Certificates: Key Factors

- To what extent Bob can trust Charlie? (usually only well-established companies play the role of CA)
- How Bob learns the public key of CA?
  - many operating systems/web browsers typically come pre-configured with many CA's public keys
- Single or Multiple CAs
- Expiration of certificates (include expiration date as part of the message before generating the signature)

Topics Covered

- Private-key cryptography
  - Perfect secrecy and one-time pad
  - Block ciphers and their modes of operation
  - MAC and Hash Functions
- Public-key cryptography
  - Number theory overview
  - Key management and exchange protocols
  - RSA
  - ElGamal
  - Digital Signatures

Practice Problems 1

- Can we achieve message integrity in MACs without private keys?
- What are the disadvantages of DES and 2DES (or why these schemes are not used in practice)?
- If x is an element of $\mathbb{Z}_N$, what is the condition on x to have an inverse?
- What is the inverse of 3 in $\mathbb{Z}_7$?
- When can you say that an element x is a generator of $\mathbb{Z}_p^*$

Practice Problems 2

- What are the issues associated in using KDCs for distributing private keys?
- Explain the encryption and decryption process under RSA
- Explain the encryption and decryption process under ElGamal
- Explain the differences between MAC and digital signatures
- Why do we hash before generating a digital signature for a message?
- Describe the importance of digital certificates?

Some Tips

- Go through everything covered in the course
- Majority of the questions will be from the topics covered after midterm exam
- Clearly understand the concepts/techniques discussed
- Solve the examples and sample problems given in the lecture slides

Best of Luck :)