CSIT 495/595 - Introduction to Cryptography
Midterm Exam Review

Bharath K. Samanthula
Department of Computer Science
Montclair State University

Topics Covered

- Traditional ciphers
- Perfect secrecy and one-time pad
- Block ciphers and their modes of operation
- Message authentication codes
    - Authenticated encryption and its types
    - Secure communication sessions
- Hash functions and HMAC

What is Cryptography?

- Cryptography is the art of writing or solving codes
- Cryptography deals with *the scientific study of techniques for securing digital information, transactions, and distributed computations* (Textbook)
- Applications:
  - Secure communication (e.g., HTTPS in Email, e-commerce, ATM machines or cellular phones)
  - Encrypting files on Disk (EFS, TrueCrypt)
  - Content Protection (e.g., Blue-Ray, DVD): CSS, AACS
  - User Authentication (e.g., verifying user password/identity)
  - .... and many other electronic applications over Internet

Three Crucial Goals

- Data Confidentiality: use encryption to prevent an adversary from learning anything about the content of the messages transmitted over an open communication channel
- Message Integrity: how a party receiving a message can be sure that it was sent by the claimed sender and was not modified in transit
- Data Authentication: Only authenticated people can receive/send messages in communication

Private-Key Encryption: Introduction

- Classical cryptosystems are mostly based on private-key encryption where the security depends on a secret, commonly called key
- General setting: two parties sharing a key can communicate and exchange messages using the key
- The eavesdropper can monitor all the communication between the two parties
- Basic Steps:
  - Party 1 encrypts a message (known as plaintext) using the shared key
  - Party 1 sends the encrypted message (known as ciphertext) to party 2
  - Party 2 decrypts ciphertext to get the actual message
- Also referred to as symmetric-key setting

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 $c \leftarrow \mathsf{Enc}_k(m)$, where $c \in \mathcal{C}$

Decryption (Dec)

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

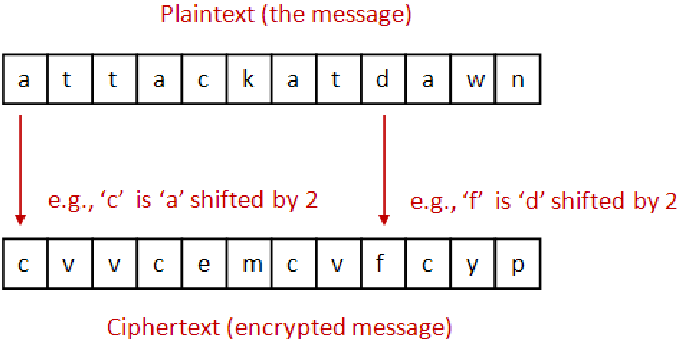
Kerckhoff's Principle

- 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

Caesar's Cipher - Example



- Note: punctuation, spaces, and numbers are removed
- What is the ciphertext when Caesar's cipher is used?

Shift Cipher - Example

Let $k = 20$

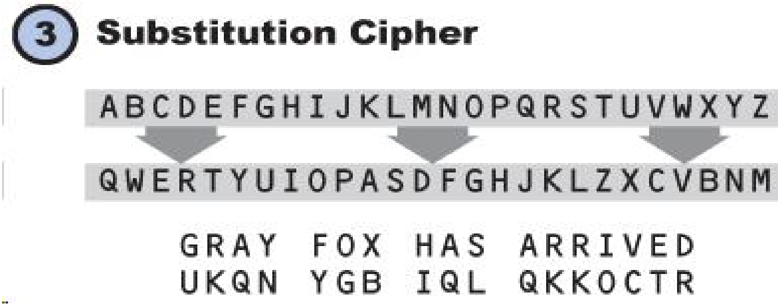
Original	M	A	T	H	R	U	L	E	S
Number-fied	12	0	19	7	17	20	11	4	18
+key	32	20	39	27	37	40	31	24	38
mod 26	6	20	13	1	11	14	5	24	12
Letter-fied	G	U	N	B	L	O	F	Y	M

Substitution Ciphers

- Mono-alphabetic
- Poly-alphabetic

Mono-alphabetic Substitution Cipher

- The key defines a fixed substitution for individual letters in the plaintext
- Each letter is mapped to one of the remaining letters
- key space: consists of all the bijections or permutations



Vigenere Cipher

- A poly-alphabetic shift cipher
- Applies several independent instances of shift cipher in sequence
- Example 1:
  Plaintext: attackatdawn
  Key: lemonlemonle
  Ciphertext: lxfopvefrnhr
- Example 2:
  Plaintext: tellhimaboutme
  Key: cafecafecafeca
  CIphertext: ??

Security Guarantee - Some Thoughts

What is a good security guarantee??

- Impossible for an attacker to recover the key
- Impossible for an attacker to recover the entire plaintext
- Impossible for an attacker to recover any character of the plaintext
- Right Answer: Attacker should not know any information about the plaintext other than what he has already known

Threat Models

In order of increasing power of attacker:

- Ciphertext-only attack
- Known-plaintext attack
- Chosen-plaintext attack
- Chosen-ciphertext attack

Perfect Secrecy Concepts

- Computational Security vs. Unconditional Security
- Definition of Perfect Secrecy
- One-Time pad and its limitations
- Shannon's Theorem

Computational Security: Formal Definition

- Suppose m_0 and m_1 be two messages of same length

A private-key encryption scheme $\Pi = \langle \text{Gen, Enc, Dec} \rangle$ is computationally secure if for every polynomial-time algorithm $\mathcal{A}$, n-bit key, there exists a negligible function ϵ, such that

$$|Pr[Out(\mathrm{A}_\Pi(n,0)) = 1] - Pr[Out(\mathrm{A}_\Pi(n,1)) = 1]| \leq \epsilon(n)$$

where $Out(\mathrm{A}_\Pi(n,b)))$ denote the output bit of the experiment being run to find out that the encrypted message is m_b and $b \in [0,1]$

Pseudorandom Generator

- A pseudorandom generator G is an efficient, deterministic algorithm:
  - Input: a short, uniform string, called the seed
  - Ouput: a longer, uniform looking string
- Cryptographic schemes are impossible without pseudorandom generators
- They are used often in
  - generating keys
  - initialization vectors
  - public-key cryptosystems
  - other cryptographic algorithms

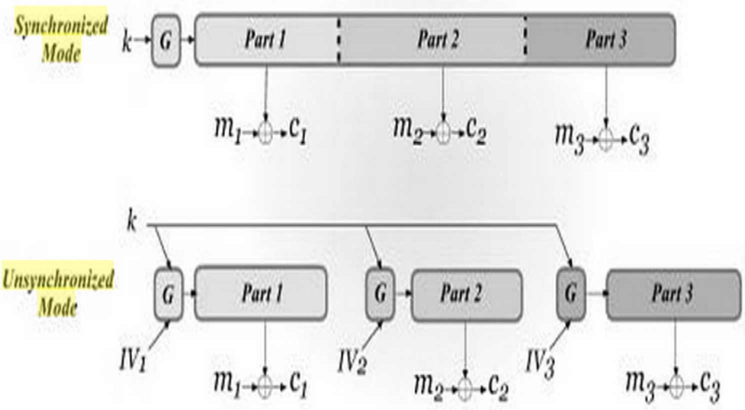
Security Under Chosen-Plaintext Attacks (CPA)

- Key Idea: encryption of a plaintext should completely yield different ciphertexts
- Nowadays, security against CPA is the minimal notion of security an encryption scheme should satisfy
- Is one-time pad encryption scheme considered CPA-secure?

Stream Ciphers

- Used for encrypting streamed data
- Encryption/decryption is done one bit at a time
- Usually faster and have a lower hardware complexity
- Used for cell phones or small embedded devices (e.g., A5/1 stream cipher is used as a standard in GSM mobile phones for voice encryption)
- Basic Idea:
  - Keystream: a pseudorandom sequence of bits, $s_1, s_2, \ldots, s_\ell$
  - Encryption: $c_i \leftarrow m_i + s_i \bmod 2 = m_i \oplus s_i$
  - Decryption: $m_i \leftarrow c_i + s_i \bmod 2 = c_i \oplus s_i$

Stream Ciphers: Modes of Operation



where IV denotes the initialization vector or nonce

Block Ciphers

- Encryption is done block by block
- Each block typically consists of 64-bit or more
- Encrypts each block with the same key
- Some well-known block ciphers - DES, AES (more details on these later)
- Basic Idea:
  - Divide the message into blocks (each of equal size)
  - If text in a block is less than its size, use padding
  - Choose the mode of operation
  - Apply the mode of operation on the blocks

Block Ciphers: Modes of Operation

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

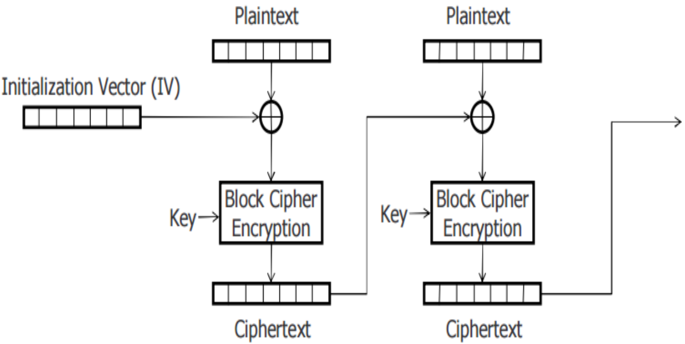
Block Ciphers: Evaluation Criteria

- Identical messages
  - under which conditions ciphertext of two identical messages are the same
- Chaining dependencies
  - how adjacent plaintext blocks affect encryption of a plaintext block
- Error propagation
  - resistance to channel noise
- Efficiency
  - preprocessing
  - parallelization: random access

Electronic Code Book (ECB) Mode

- Direct use of the block cipher/ pseudorandom functions
- Apply block cipher to each plaintext block
- Used primarily to transmit encrypted keys
- Never use it for general-purpose encryption, such as for a file or a image (why??)

Cipher Block Chaining (CBC) Mode

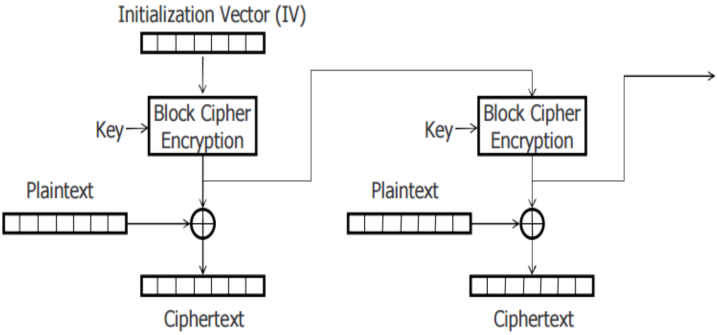


- Allows random access to ciphertext
- Decryption is parallelizable
  - Plaintext block x_j requires ciphertext blocks c_j and c_{j-1}

Cipher Block Chaining (CBC) Mode

- Identical messages: changing IV or the first plaintext block results in different ciphertext
- Chaining: Ciphertext block c_j depends on x_j and all preceding plaintext blocks (dependency contained in c_{j-1})
- Error propagation: Single bit error on c_j may flip the corresponding bit on x_{j+1}, but changes x_j significantly.
- IV need not be secret, but its integrity should be protected

Output Feedback (OFB) Mode

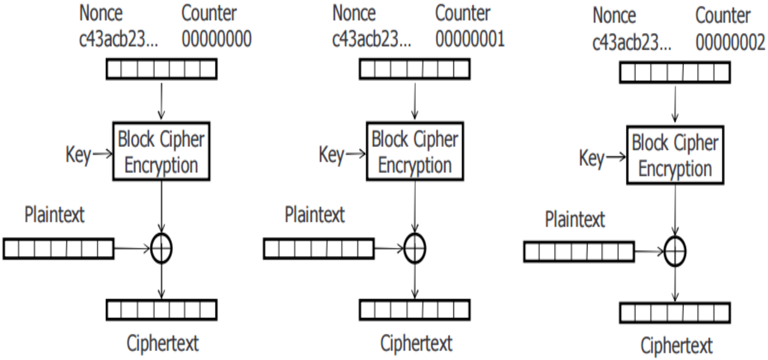


- Preprocessing possible (keep enc/decrypting previous output block)
- No random access, not parallelizable

Output Feedback (OFB) Mode

- Identical messages: same as CBC
- No chaining dependencies
- Error propagation: Single bit error on c_j may only affect the corresponding bit of x_j
- IV need not be secret, but should be changed if a previously used key is to be used again

Counter (CTR) Mode



- Preprocessing possible (inc/decrement and enc/decrypt counter)
- Allows random access

Counter (CTR) Mode

- Both encryption & decryption are parallelizable
  - Encrypted counter is sufficient to enc/decrypt
- Identical messages: changing nonce results in different ciphertext
- No chaining dependencies
- No error propagation
- Nonce should be random, and should be changed if a previously used key is to be used again

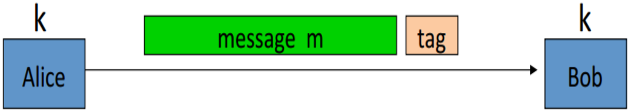
Data Confidentiality vs. Message Integrity

- 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 ← S(k, m)

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

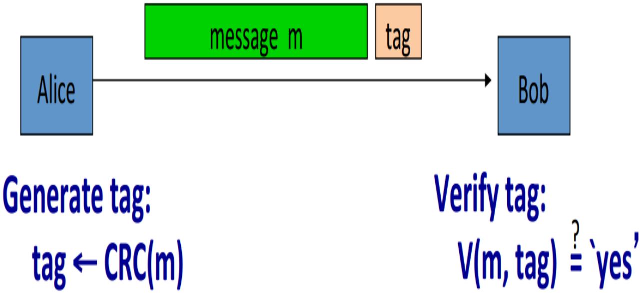
Def: **MAC** I = (S,V) defined over (K,M,T) is a pair of algs:

- 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?



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

MAC Security (1)

Attacker's power: **chosen message attack**

- for $m_1, m_2, \ldots, m_q$ attacker is given $t_i \leftarrow S(k, m_i)$

Attacker's goal: **existential forgery**

- produce some **new** valid message/tag pair (m,t).

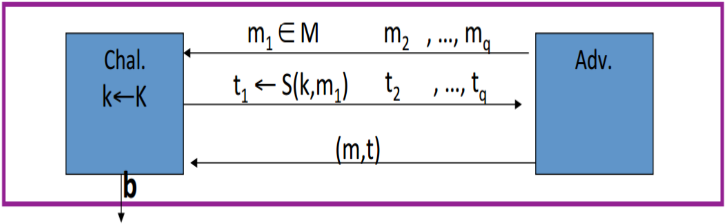
$$(m,t) \notin \{ (m_1,t_1), \ldots, (m_q,t_q) \}$$

$\Rightarrow$ attacker cannot produce a valid tag for a new message

$\Rightarrow$ given (m,t) attacker cannot even produce (m,t') for $t' \neq t$

MAC Security (2)

- For a MAC $I=(S,V)$ and adv. A define a MAC game as:



$$\begin{cases} \mathbf{b}=1 & \text{if } V(k,m,t) = \text{`yes'} \text{ and } (m,t) \notin \{(m_1,t_1), \dots, (m_q,t_q)\} \\ \mathbf{b}=0 & \text{otherwise} \end{cases}$$

Def: $I=(S,V)$ is a **secure MAC** if for all "efficient" A:

$$\mathrm{Adv}_{\mathrm{MAC}}[A,I] = \Pr[\text{Chal. outputs } 1] \quad \text{is "negligible."}$$

MAC - Sample Question

Let I = (S,V) be a MAC.

Suppose S(k,m) is always 5 bits long

Can this MAC be secure?

- ○ No, an attacker can simply guess the tag for messages
- ○ It depends on the details of the MAC
- ○ Yes, the attacker cannot generate a valid tag for any message

MAC Example: Protecting System Files

Suppose at install time the system computes:

filename F_1	filename F_2	…	filename F_n
$t_1 = S(k, F_1)$	$t_2 = S(k, F_2)$		$t_n = S(k, F_n)$

k derived from user's password

Later a virus infects system and modifies system files

User reboots into clean OS and supplies his password

– Then: secure MAC $\Rightarrow$ all modified files will be detected

Secure PRF $\rightarrow$ Secure MAC: A Bad Example

Suppose $F: K \times X \longrightarrow Y$ is a secure PRF with $Y = \{0,1\}^{10}$

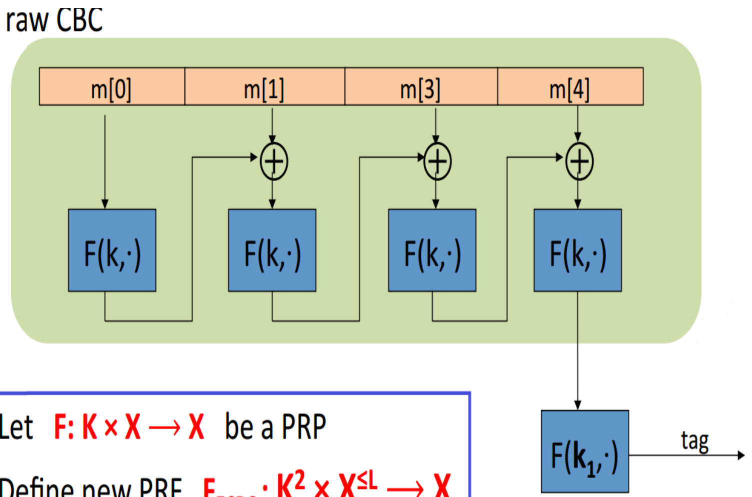
Is the derived MAC I_F a secure MAC system?

- Yes, the MAC is secure because the PRF is secure
- No tags are too short: anyone can guess the tag for any msg
- It depends on the function F

CBC-MAC

- Similar to CBC-mode encryption, except with the following two differences.
  - IV is a random vector in CBC-mode encryption whereas there is no IV in CBC-MAC
  - In CBC-mode encryption, multiple ciphertexts (one for each block) are output whereas in CBC-MAC there is only one output from the final block

Encrypted CBC-MAC (ECBC-MAC)



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

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

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

Encrypt-and-authenticate

- Given a message m, the sender transmits (c, t) to the receiver where

$$c \leftarrow Enc_{k_E}(m) \quad \text{and} \quad t \leftarrow S(k_M, m)$$

- The receiver decrypts c, and verifies the tag t
- Limitations:
  - tag $S(k_M, m)$ can leak information to eavesdropper
  - Example: For a MAC where the first bit of the tag is always equal to the first of the message
  - If a deterministic MAC like CBC-MAC is used, then the tag remains the same for a given message and key k_M

Authenticate-then-encrypt

- Given a message m, the sender computes the ciphertext c as follows

$$t \leftarrow S(k_M, m) \quad \text{and} \quad c \leftarrow Enc_{k_E}(m \| t)$$

- The receiver decrypts c to obtain $m \| t$ and verifies the tag t
- Limitations:
  - Two error messages possible: "bad padding" and "authentication failure"
  - If the attacker can distinguish between the two errors, he/she can recover the whole plaintext from a given ciphertext
  - A real-world attack: in configurations of IPsec

Encrypt-then-authenticate

- Given a message m, the sender computes (c, t) as follows

$$c \leftarrow Enc_{k_E}(m) \quad \text{and} \quad t \leftarrow S(k_M, c)$$

- Receiver first verifies t. If successful, then he decrypts c
- Observations:
  - This approach is sound, as long as the MAC is strongly secure
  - MAC is verified before decryption takes place, so MAC verification process cannot leak anything about the plaintext

Secure Communication Session

- Often parties wish to communicate securely (that is achieving both secrecy and integrity) over the course of a communication session
- A naive way of encrypting the message using authenticated encryption may not work
- Potential Attacks
  - Re-ordering Attack: The attacker can swap the order of messages
  - Replay Attack: The attacker can send (replay) a valid ciphertext to Bob which was previously sent by Alice
  - Reflection Attack: An attacker can take a ciphertext c, which was earlier sent from Alice to Bob, and send it back to Alice
- The above attacks can be easily prevented using counters and a directionality bit/separate keys for parties (How ??)

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 $H: M \to T$ 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:

$$\mathrm{Adv}_{CR}[A,H] = \Pr[\text{ A outputs collision for H}]$$

is "neg".

Example: SHA-256 (outputs 256 bits)

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

Protecting File Integrity: Example

Software packages:

package name F_1	package name F_2	…	package name F_n

read-only public space

$H(F_1)$ $H(F_2)$ $H(F_n)$

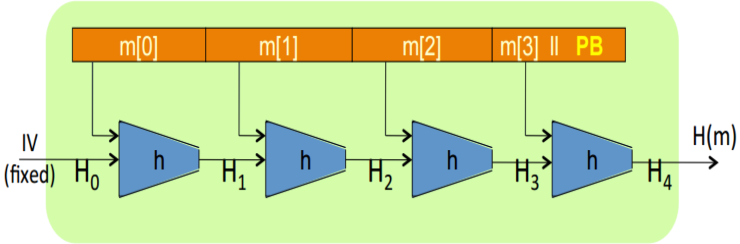
When user downloads package, can verify that contents are valid

H collision resistant $\Rightarrow$

attacker cannot modify package without detection

no key needed (public verifiability), but requires read-only space

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 $\boxed{1000\ldots0 \ \| \ \underbrace{\text{msg len}}_{\text{64 bits}}}$

If no space for PB add another block

Dan Boneh

MAC construction from Hash Functions

Can we use $H(\cdot)$ to directly build a MAC?

$H: X^{\leq L} \to T$ a C.R. Merkle-Damgard Hash Function

Attempt #1: $S(k, m) = H(k \parallel m)$

This MAC is insecure because:

- ○ Given $H(k \parallel m)$ can compute $H(w \parallel k \parallel m \parallel PB)$ for any w.
- ○ Given $H(k \parallel m)$ can compute $H(k \parallel m \parallel w)$ for any w.
- ○ Given $H(k \parallel m)$ can compute $H(k \parallel m \parallel PB \parallel w)$ for any w.
- ○ Anyone can compute $H(k \parallel m)$ for any m.

Standardized Method: Hash-MAC (HMAC)

Most widely used MAC on the Internet.

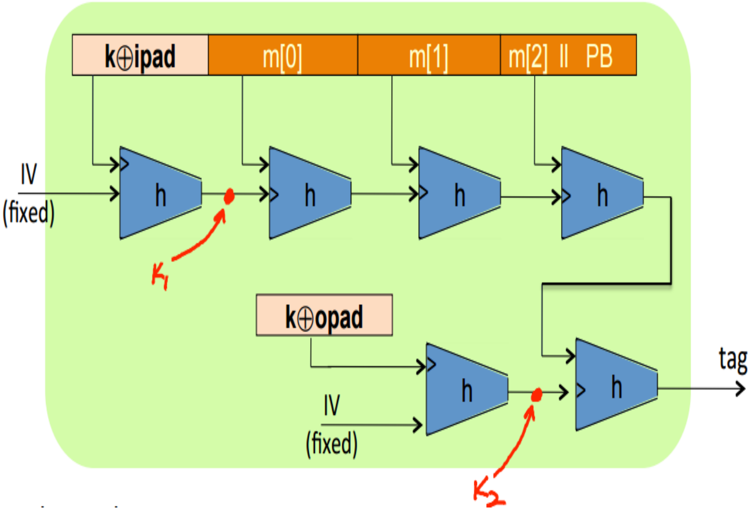
H: hash function.

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



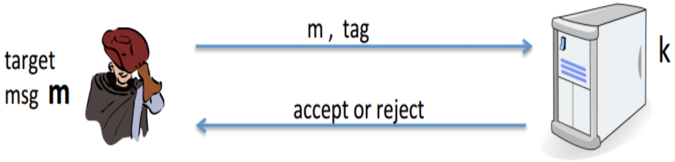
Hash Functions: Generic Attacks

Birthday Attack

- Birthday Problem - Given q people in a room, what is the probability that two of them have the same Birthday
- For the Birthday problem, when $q = \Theta(N^{1/2})$, the probability for two of them have the same birthday is greater than 1/2 (where $N = 365$)
- For hash functions, given q distinct inputs $x_1, \ldots, x_q$ and their hash values, q should be at least $\Theta(2^{\ell/2})$ to achieve collision probability of roughly 1/2
- Example: to make finding hash collisions as difficult as an exhaustive search over 128-bit keys, the output length should be at least 256 bits

Hash Functions: Generic Attacks

MAC Timing attacks



Timing attack: to compute tag for target message m do:

Step 1: Query server with random tag

Step 2: Loop over all possible first bytes and query server.

stop when verification takes a little longer than in step 1

Step 3: repeat for all tag bytes until valid tag found

How can we avoid timing atttacks?

Some Applications of Hash Functions

- Fingerprinting and Deduplication
  - Virus Fingerprinting
  - Deduplication
  - (Peer-to-peer) P2P file sharing
- Merkle Trees
- Password Hashing
- ... and many others

Topics Covered

- Traditional ciphers
- Perfect secrecy and one-time pad
- Block ciphers and their modes of operation
- Message authentication codes
    - Authenticated encryption and its types
    - Secure communication sessions
- Hash functions and HMAC

More Practice Problems

- What is the effect of a single-bit error in the ciphertext when using different modes of operation?
- Define the goals of authenticated encryption and its constructions.
- When can we say a MAC scheme is secure?
- How can we construct a secure MAC from hash functions?
- If H is a collision resistant hash function, would $H'(m) = H(|m|)$ is collision resistant too?

Some Tips

- Understand the concepts/techniques discussed
- Solve the examples and sample problems given in the lecture slides

Best of Luck :)