Message Authentication Code and Digital Certificate

Kehuan Zhang

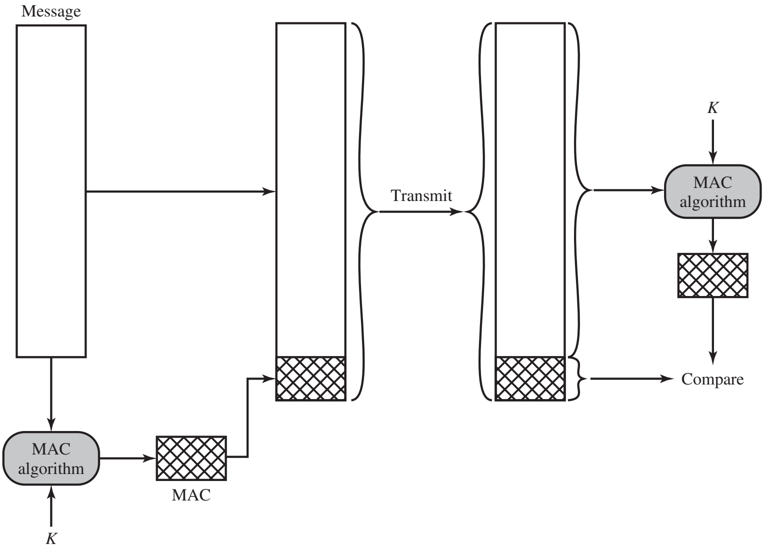
IERG4130 2022

Message Authentication Code and Hash Function

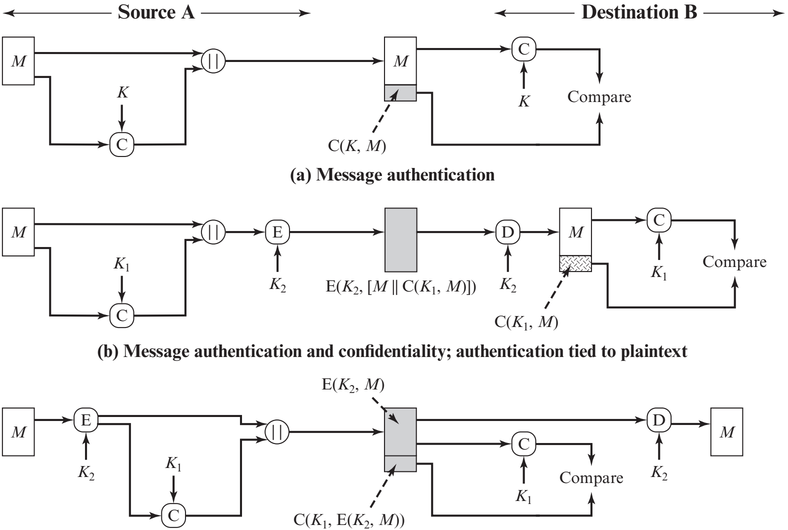
What is Message Authentication?

- Procedure that allows communicating parties to verify that received messages are authentic, namely
  - source is authentic – not from masquerading
  - contents unaltered – message has not been modified
  - timely sequencing – the message is not a replay of a previously sent one
- Message Authentication Code (MAC) can achieve above goals
  - Receiver assured that message is not altered – no modification
  - Receiver assured that the message is from the alleged sender – no masquerading
  - Include a sequence number, assured proper sequence – no replay

Operations of Message Authentication Code



Combine Integrity with Confidentiality



(a) Message authentication

(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

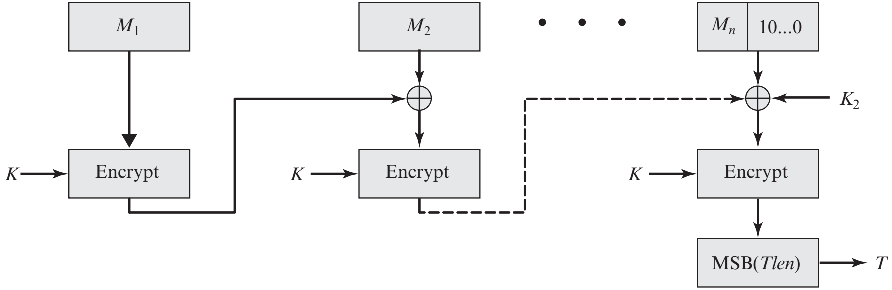
Ways to provide Message Authentication

- Message Authentication via Conventional Encryption
  - Only the sender and receiver should share a key ;
  - Include a time-stamp or "nonce" to prevent replay attack
  - Implicitly assume the receiver can recognize if the output from decryption unit is garbage or not
    - ★ Easy if they know the message has some specific format, e.g. English
    - ★ May be difficult if the original plaintext are random binary data
    - ★ `Then other measures are needed, e.g., checksum`
- Message Authentication without Message Encryption
  - Thus no message confidentiality
  - An authentication tag (aka Message Authentication Code or MAC) is generated and appended to each message where
    - ★ The MAC is computed as a publicly known function F, of the message M and a shared secret key K:

$$MAC = F(K, M)$$

  - A one-way Hash function can be used as F

MAC based on CBC-Residue



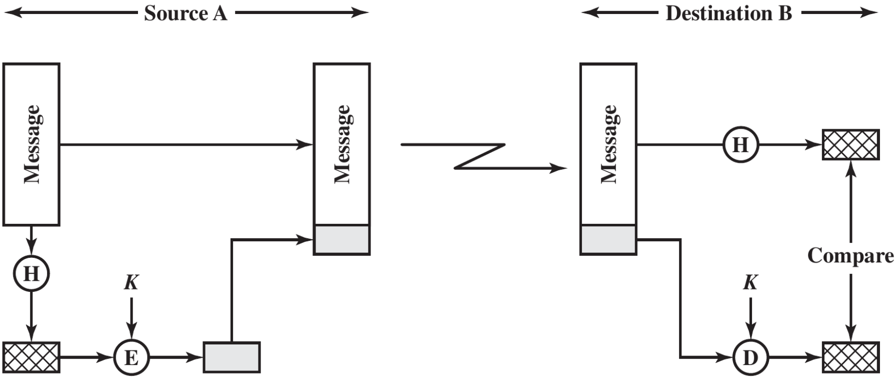
MAC based on CBC-Residue (Cont.)

- The last encrypted block, aka the CBC residue, can be used as a "Message Authentication Code" (MAC) for a message as follows:
  1. The sender transmits the original message in plaintext together with the the CBC residue (but NOT the key, of course)
  2. The receiver, who knows the key in advance, can then encrypt the plaintext upon its arrival using CBC mode. If the message has been tampered with during transmission, the CBC residue won't match !
- Notice in this case, CBC is used for MAC purpose and does NOT provide secrecy at all ;
  - If both secrecy and message-authenticity (tamper-proof) are required, we need to do CBC twice in 2 passes with 2 different keys:
    - ★ 1st pass for encryption,
    - ★ 2nd pass to generate the CBC-residue for MAC.
  - Why is it insufficient to just append the CBC residue of the 1 st pass as the MAC ?

Drawbacks of Encrypted-based MAC

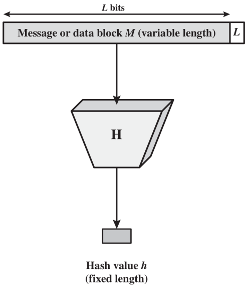
- Encryption software is slow
- Encryption hardware costs aren't cheap
- Hardware optimized toward large data sizes
- Encryption Algorithms are usually covered by patents
- Algorithms subject to US export control

MAC based on One-way Hash Function



One-Way Hash Function

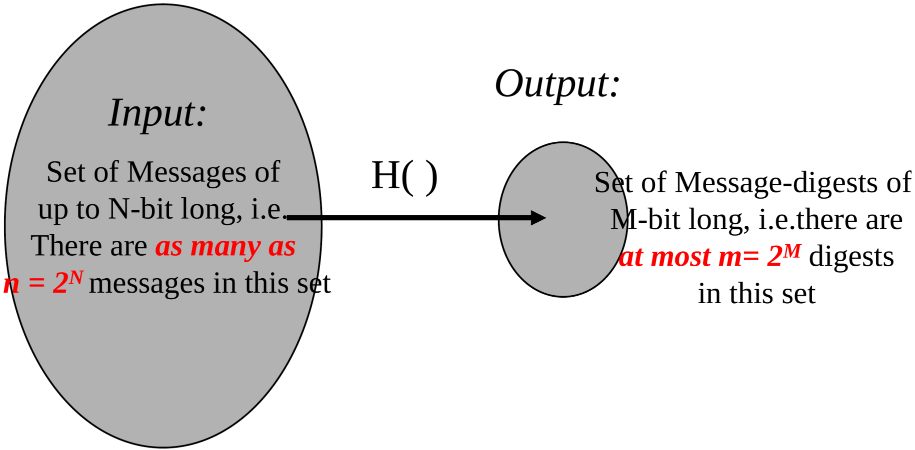
- Hash function accepts a variable size message M as input and
- produces a fixed-size message digest H(M) as output
- Message digest is sent with the message for authentication
- Produces a fingerprint of the message
- No secret key is involved



Requirements on One-Way Hash Functions $H(\cdot)$

1. $H(\cdot)$ can be applied to a block of data of any size
2. $H(\cdot)$ produces a fixed length output
3. $H(x)$ is relatively easy to compute
4. For any given hashed value h, it is computationally infeasible to find an input x such that $H(x) = h$
   - i.e. safe against the so-called *1st preimage attack*
   - guarantee the property of "one-way"
5. For any given message x, it is computationally infeasible to find another message $y \neq x$ with $H(y) = H(x)$
   - i.e. safe against the so-called *2nd preimage attack*
   - Weak collision resistance
6. It is computationally infeasible to find any pair of message (x, y) such that $H(x) = H(y)$ (i.e., having the same hashed value)
   - Strong collision resistance (e.g., agaist **birthday attack**)

How Likely to have Hash Output Collisions?



Since $N >> M$, (and therefore) $n >> m$, collisions are inevitable no matter how secure the one-way function $H(\cdot)$ is.

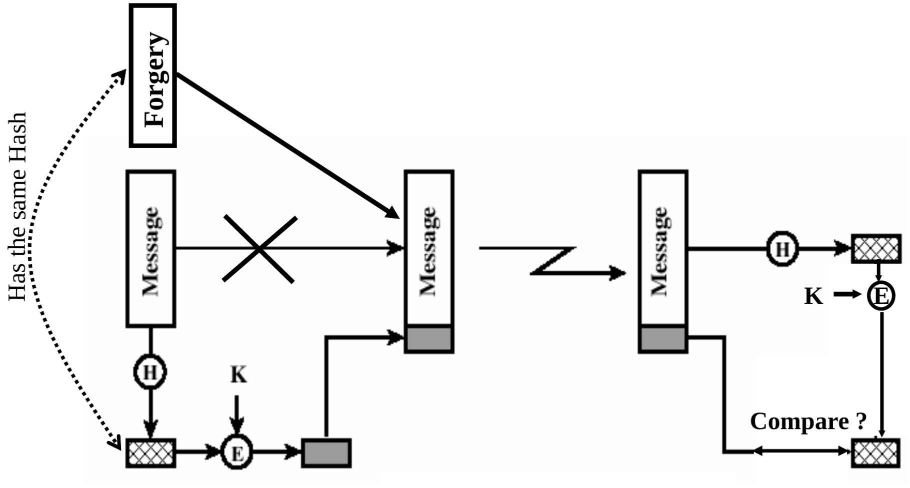
The Birthday Paradox

- In a room with n people, what is the probability $P_{collision}$ that we will find at least 2 people who have the same birthday
  - Assuming birthdays are uniformly distributed (with m=365 days)
  - $P_{no\_collision} = \overbrace{\frac{m}{m} \cdot \frac{m-1}{m} \cdots \frac{m-(n-1)}{m}}^{n} = \frac{m\cdot(m-1)\cdot(m-2)\cdot\cdots\cdot(m-n+1)}{m^n}$
$$= \frac{m^n + (-1-2\cdots-(n-1))m^{n-1} + O(m)}{m^n}$$
$$= 1 - \frac{\frac{(n-1)((n-1)+1)}{2} m^{n-1}}{m^n} + \frac{O(m)}{m^n} = 1 - \frac{n(n-1)}{2m} + \frac{O(m)}{m^n}$$
  - If we require $P_{collision} > \frac{1}{2}$, then $P_{collision} = 1 - P_{no\_collision} = 1 - (1 - \frac{n(n-1)}{2m} + \frac{O(m)}{m^n}) = \frac{n(n-1)}{2m} - \frac{O(m)}{m^n} > \frac{1}{2}$, since $\frac{O(m)}{m^n} > 0$, $\frac{n(n-1)}{2m} > \frac{1}{2} + \frac{O(m)}{m^n} > \frac{1}{2}$, so $n > \sqrt{m+n} > \sqrt{m}$
  - When $m = 365, P_{collision} > \frac{1}{2}$ when $n >= 20$
    - ★ Which means: if there are 20 people in a room, it is very likely ($Pr > 0.5$) that we will find at least 2 people who have the same birthday

How difficult to find a Hash collision ?

- How secure is a one-way hash with 64-bit output, e.g. DES in CBC mode ?
- Based on the property of a good hash function, the hash output of any input string should be uniformly distributed over the hash output space of size $m = 2^{64}$
  - This is analogous to the fact that the birthday of any given person is uniformly distributed over any days within a year (i.e. output space of size m = 365)
  - Thus, according to the Birthday Paradox, if no. of all possible outcomes = m, we only need to try about n = $\sqrt{m}$ inputs to the hash function to have a good chance to find a collision,
    - ★ e.g. For, a hash function with 64-bit output, $m = 2^{64}$
    - ★ => it only takes about $\sqrt{m} = 2^{32}$ tries to find a pair of inputs which will produce the same hash output, i.e. a collision

Birthday Attack on Hash-based Message Digest



Birthday Attack on Hash-based Message Digest (Cont.)

- Birthday attack can proceed as follows: opponent generates 2^{32} variations of a valid message, all with essentially the same meaning ; this is " doable" given current technology.
- opponent also generates 2^{32} variations of a desired fraudulent message
- two sets of messages are compared to find a pair with same hash output (by argument similar to the Birthday paradox, this probability > 0.5)
- have user (the victim) sign the valid message, but sent the forgery message which will have a valid message digest
- Conclusion is that we need to use longer MACs
- BTW, how can we generate 2^{32} variations of a letter carrying the same meaning ?
  - Just 2 choices of wording at 32 different places.

An Example of Birthday Attack

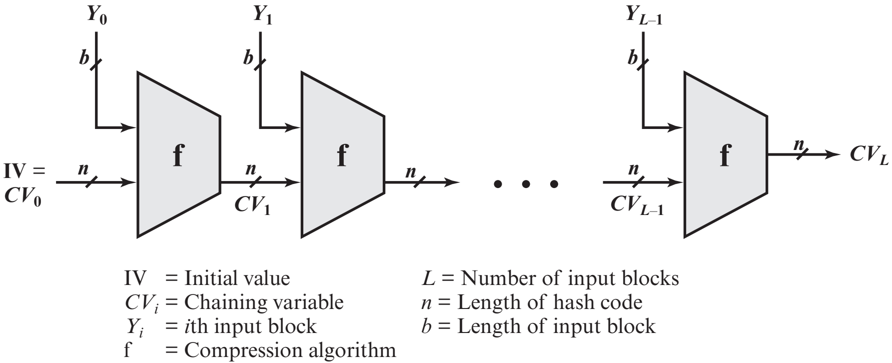
As $\begin{Bmatrix}\text{the}\\ \text{—}\end{Bmatrix}$ Dean of Blakewell College, I have $\begin{Bmatrix}\text{had the pleasure of knowing}\\ \text{known}\end{Bmatrix}$ Cherise Rosetti for the $\begin{Bmatrix}\text{last}\\ \text{past}\end{Bmatrix}$ four years. She $\begin{Bmatrix}\text{has been}\\ \text{was}\end{Bmatrix}$ $\begin{Bmatrix}\text{a tremendous}\\ \text{an outstanding}\end{Bmatrix}$ $\begin{Bmatrix}\text{asset to}\\ \text{role model in}\end{Bmatrix}$ $\begin{Bmatrix}\text{our}\\ \text{the}\end{Bmatrix}$ school. I $\begin{Bmatrix}\text{would like to take this opportunity to}\\ \text{wholeheartedly}\end{Bmatrix}$ recommend Cherise for your $\begin{Bmatrix}\text{school's}\\ \text{—}\end{Bmatrix}$ graduate program. I $\begin{Bmatrix}\text{am}\\ \text{feel}\end{Bmatrix}$ $\begin{Bmatrix}\text{confident}\\ \text{certain}\end{Bmatrix}$ $\begin{Bmatrix}\text{that}\\ \text{—}\end{Bmatrix}$ $\begin{Bmatrix}\text{she}\\ \text{Cherise}\end{Bmatrix}$ will $\begin{Bmatrix}\text{continue to}\\ \text{—}\end{Bmatrix}$ succeed in her studies. $\begin{Bmatrix}\text{She}\\ \text{Cherise}\end{Bmatrix}$ is a dedicated student and $\begin{Bmatrix}\text{thus far her grades}\\ \text{her grades thus far}\end{Bmatrix}$ $\begin{Bmatrix}\text{have been}\\ \text{are}\end{Bmatrix}$ $\begin{Bmatrix}\text{exemplary}\\ \text{excellent}\end{Bmatrix}$. In class, $\begin{Bmatrix}\text{she}\\ \text{Cherise}\end{Bmatrix}$ $\begin{Bmatrix}\text{has proven to be}\\ \text{has been}\end{Bmatrix}$ a take-charge $\begin{Bmatrix}\text{person}\\ \text{individual}\end{Bmatrix}$ $\begin{Bmatrix}\text{who is}\\ \text{—}\end{Bmatrix}$ able to successfully develop plans and implement them.

$\begin{Bmatrix}\text{She}\\ \text{Cherise}\end{Bmatrix}$ has also assisted $\begin{Bmatrix}\text{us}\\ \text{—}\end{Bmatrix}$ in our admissions office. $\begin{Bmatrix}\text{She}\\ \text{Cherise}\end{Bmatrix}$ has $\begin{Bmatrix}\text{successfully}\\ \text{—}\end{Bmatrix}$ demonstrated leadership ability by counseling new and prospective students. $\begin{Bmatrix}\text{Her}\\ \text{Cherise's}\end{Bmatrix}$ advice has been $\begin{Bmatrix}\text{a great}\\ \text{of considerable}\end{Bmatrix}$ help to these students, many of whom have $\begin{Bmatrix}\text{taken time to share}\\ \text{shared}\end{Bmatrix}$ their comments with me regarding her pleasant and $\begin{Bmatrix}\text{encouraging}\\ \text{reassuring}\end{Bmatrix}$ attitude. $\begin{Bmatrix}\text{For these reasons}\\ \text{It is for these reasons that}\end{Bmatrix}$ I $\begin{Bmatrix}\text{highly recommend}\\ \text{offer high recommendations for}\end{Bmatrix}$ Cherise $\begin{Bmatrix}\text{without reservation}\\ \text{unreservedly}\end{Bmatrix}$. Her $\begin{Bmatrix}\text{ambition}\\ \text{drive}\end{Bmatrix}$ and $\begin{Bmatrix}\text{abilities}\\ \text{potential}\end{Bmatrix}$ will $\begin{Bmatrix}\text{truly}\\ \text{surely}\end{Bmatrix}$ be an $\begin{Bmatrix}\text{asset to}\\ \text{plus for}\end{Bmatrix}$ your $\begin{Bmatrix}\text{establishment}\\ \text{school}\end{Bmatrix}$.

Hash Algorithms

- Basic building blocks
- Typical algorithms and their security
  - MD5 Message Digest
    - ★ Not secure any more
    - ★ $\sqrt{2^{128}} = 2^{64}$
    - ★ Reduced the complexity to $2^{24.1}$
    - ★ Collisions can be found in seconds with a modern PC!
  - SHA-1
    - ★ Attacking is feasible for attackers with enough resources
  - SHA-2
    - ★ SHA512
  - SHA-3

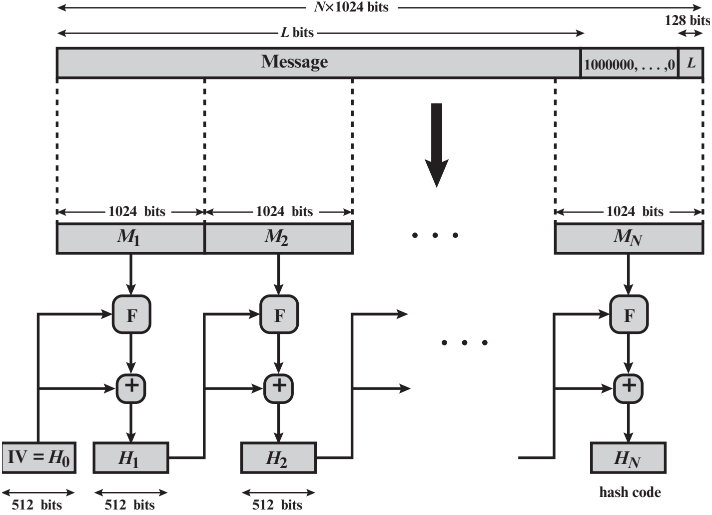
The Building Block - Merkle-Damgård Construction



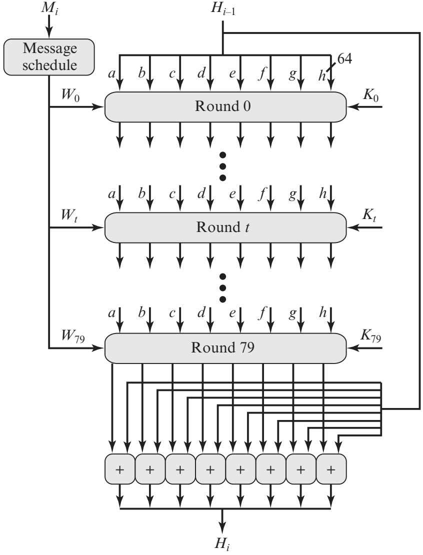
- Divide the inputs into data blocks
- Chaining structure
- The compression function $f()$ is very important

The SHA512 Algorithm

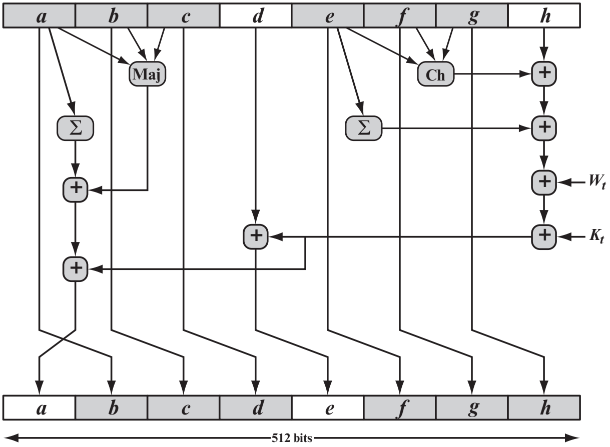
- Input block size is 1024-bit, output hash value is 512-bit



Operations for each Block in SHA512



Operations within each Round of SHA512



Hash Algorithm of SHA-1 (Secure Hash Algorithm 1)

- SHA Was designed by NIST and NSA in 1993, revised in 1995 as SHA-1
  - It also used the Merkle Damgård Construction
  - Input data block size is 512-bit
  - Output Hash value has 160-bit
- Slower than MD5 (because of more internal operations)
- More Secure than MD5, but considered insecure
  - E.g., in 2005, the same Chinese Team who broke MD5 found a way to reduce the complexity of finding SHA-1 hash collisions from 2^{80} to 2^{68}
  - And in 2011, the complexity is further reduced to 2^{61}
  - Many parties have stopped using SHA-1

SHA-2 Algorithm Family

- NIST revised SHA-1 standards which can support longer hash value lengths, including 256, 384, and 512
  - Known as SHA-256, SHA-384, and SHA-512 respectively
  - But the underlying structure, modular arithmetic and logical operations are the same

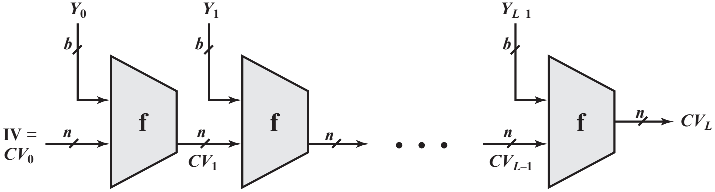
Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

SHA-3 Algorithm

- Development of SHA-3
  - In 2007, NIST initiated a competition for next generation Hash function
  - In Oct 2012, Keccak (pronounced “catchack”) was announced as the winnder
  - SHA-3 standard was officially anounced in Aug 2015 by NIST
- Does it mean SHA-2 is insecure, and we should switch to SHA-3?
  - No! Efficient attack was NOT found for SHA-2 family
- Then Why SHA-3?
  - SHA-2 algorithms are using the same structure as MD5 and SHA-1
  - Efficient attacking algorithms have been found for MD5 and SHA-1
  - How about SHA-2 family? Can the longer length really provide stronger security in the long-run?
  - SHA-3 is a hash function based on a totally different structure
    - ⋆ Sponge Construction (details will be skipped)

A Common Weakness of Merkle-Damgård Constructions (including MD5)

- Message-appending Attack (also called length extention attack)
  - If the shared secret, e.g., IV or CV_0, is used at as the initial inputs
  - Then the hash value CV_L will already contain the shared information
  - Attacks then can generate a fake message by **appending** an extra piece of message Y_L to original one
  - It allows any who add extra information to the end of original message but can still generate correct hash value (even without knowing the secret)



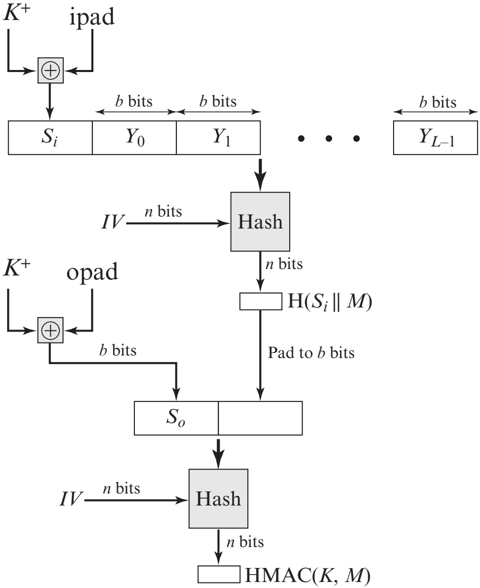
IV = Initial value
CV_i = Chaining variable
Y_i = ith input block
f = Compression algorithm

L = Number of input blocks
n = Length of hash code
b = Length of input block

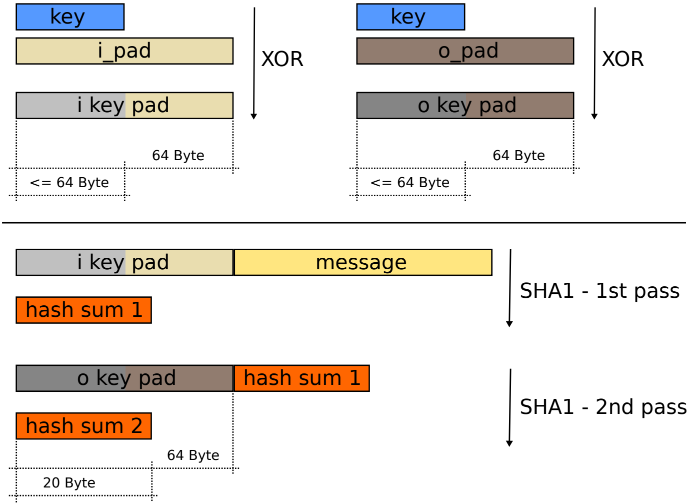
HMAC - Hash-based Message Authentication Code

- Effort to develop a MAC derived from a cryptographic hash codes such as SHA-1
- Executes faster in software
- No export restrictions
- Relies on a secret key
- RFC 2104 list design objectives and
- Provable security properties
- Used in IPsec, TLS
- Can use different digest functions as a component, e.g.
  - HMAC-SHA1, HMAC-MD5, HMAC-SHA512, etc.

HMAC Structure



More Details on HMAC



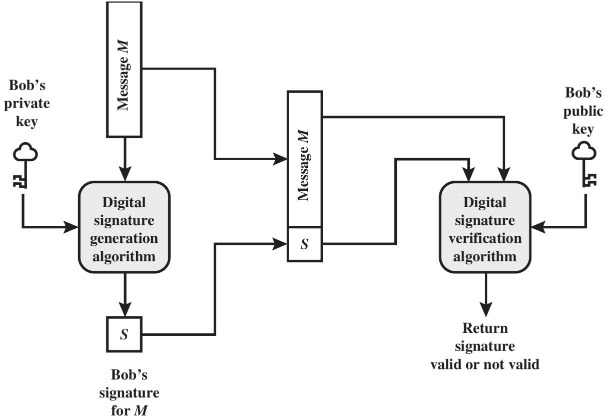
Digital Signature and Digital Certificate

Concept of Digital Signature

- Requirements on digital signature
  - Signature must depend on the document to be signed
  - Signature must use some information unique to the sender to prevent both forgery and denial
  - Relatively easy to produce and verify, but computationally infeasible to forge a signature
    - Attack 1: constructing a new message to match a given digital signature
    - Attack 2: constructing a fraudulent digital signature for a given message
- Recall the two working mode of public-key encryptions
  - Confidentiality: $c = E(K_{pub}, M) \rightarrow M = D(K_{priv}, c)$
  - Integrity: $c = E(K_{priv}, M) \rightarrow M = D(K_{pub}, c)$
    - The "integrity" mode can be used to generate **digital signature**
    - Non-repudiation, non-deniable, due to the secrecy of the private-key
- But public-key encryption is slow
  - So here comes the **Hash** function that converts message to a small and fixed length

General Model of using Digital Signature

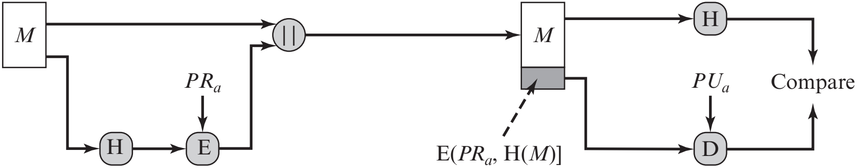
$$M; S = E(K_{priv}, Hash(M)) \rightarrow M'; Hash(M') =? D(K_{pub}, S)$$



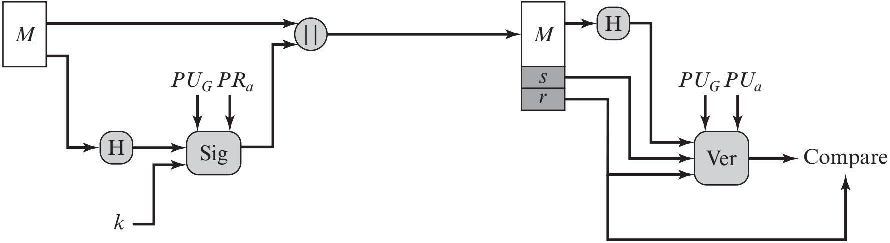
- Two key algorithms: Hash and encryption

Digital Signatures based on DSS vs. based on RSA

- RSA approach is very common
- But DSS was standardized by NIST



(a) RSA approach



(b) DSA approach

MAC vs. Digital Signature

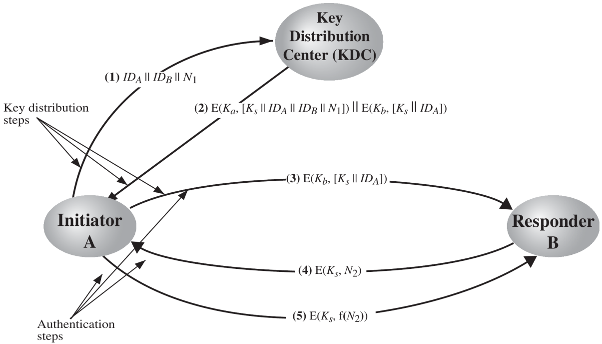
- Common things: both can be used to verify the integrity
- Differences
  - MAC cannot guarantee non-repudiation. Why?
    - ★ Essentally, it is based on symmetric key encryption (i.e., via a shared secret)
    - ★ Attackers in its adversary model is somebody in-between but not insiders
  - Digital Signature can provide guarantees on non-repudiation
    - ★ Because it is based on asymmetric key encryption

Key Management and Digital Certificate

Key Management is Important

- One major goal of public key encryption system is to solve key management problem faced by symmetric key encryption system
- But public key encryption system itself is facing a similar problem
  - Do you still remember Man-in-the-middle attacks to RSA (and Diffie-Hellman)?
  - Why is that attack feasible?
    - $\rightarrow$ unable to verify the authenticity of public keys
  - It is an chicken-or-the-egg problem
    - Verifying the authenticity needs some shared secret
    - But public key encryption does not require any previous shared secret, otherwise we were going back to symmetric encryption
- Solution? $\rightarrow$ Have to rely on a trusted 3rd party
  - Approach one: Key Distribution Center
    - Will help to generate a session key between two parties
  - Approach two: Certificate Authority (**CA**)
    - Will sign on (or generate a certificate for) a public key to guarantee its authenticity

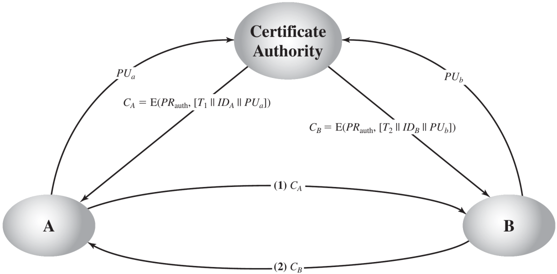
Key Distribution Center for Symmetric Key Encryption



- Terms:
  - ID_A: A's identity, K_a: shared secret between A and KDC
  - N_1 and N_2 are random nonce (to defense against replay attack)
  - K_s is the session key (between A and B)

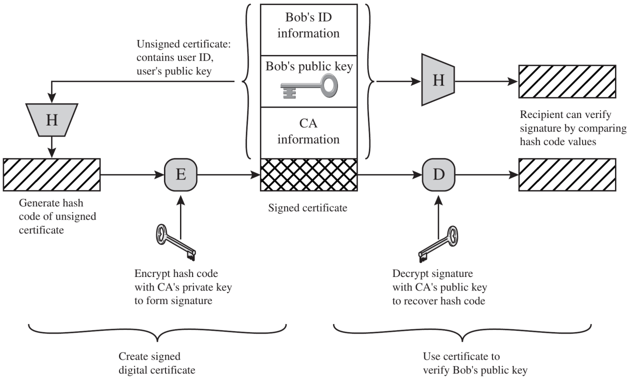
Public Key Infrastructure for Asymmetric Key Encryption

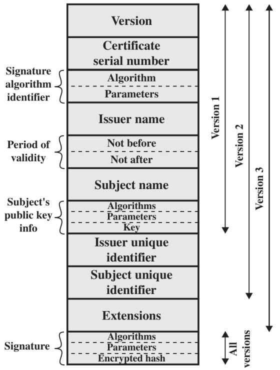
- PKI is used to manage digital certificate and public key encryptions
  - Creation, distribute, store, revoke, etc.



- PU_a and PU_b are the public key of A and B respectively
- C_A and C_B are the digital certificate of A and B respectively
- PR_{auth} is the private key of CA, ID_A: A's identity
- T is time-stamp (to prevent replaying attack)

Digital Certificate: Creation, Verification, and Structure





Summary

- In this lecture we have introduced:
  - Concepts of Message Authentication Code (MAC)
  - MAC based on CBC-residue and MAC based on Hash function
  - Requirement of Hash function and the Birthday Attack
  - High level structure of typical Hash algorithms
  - Concepts of Digital Signature
  - Key Management and PKI