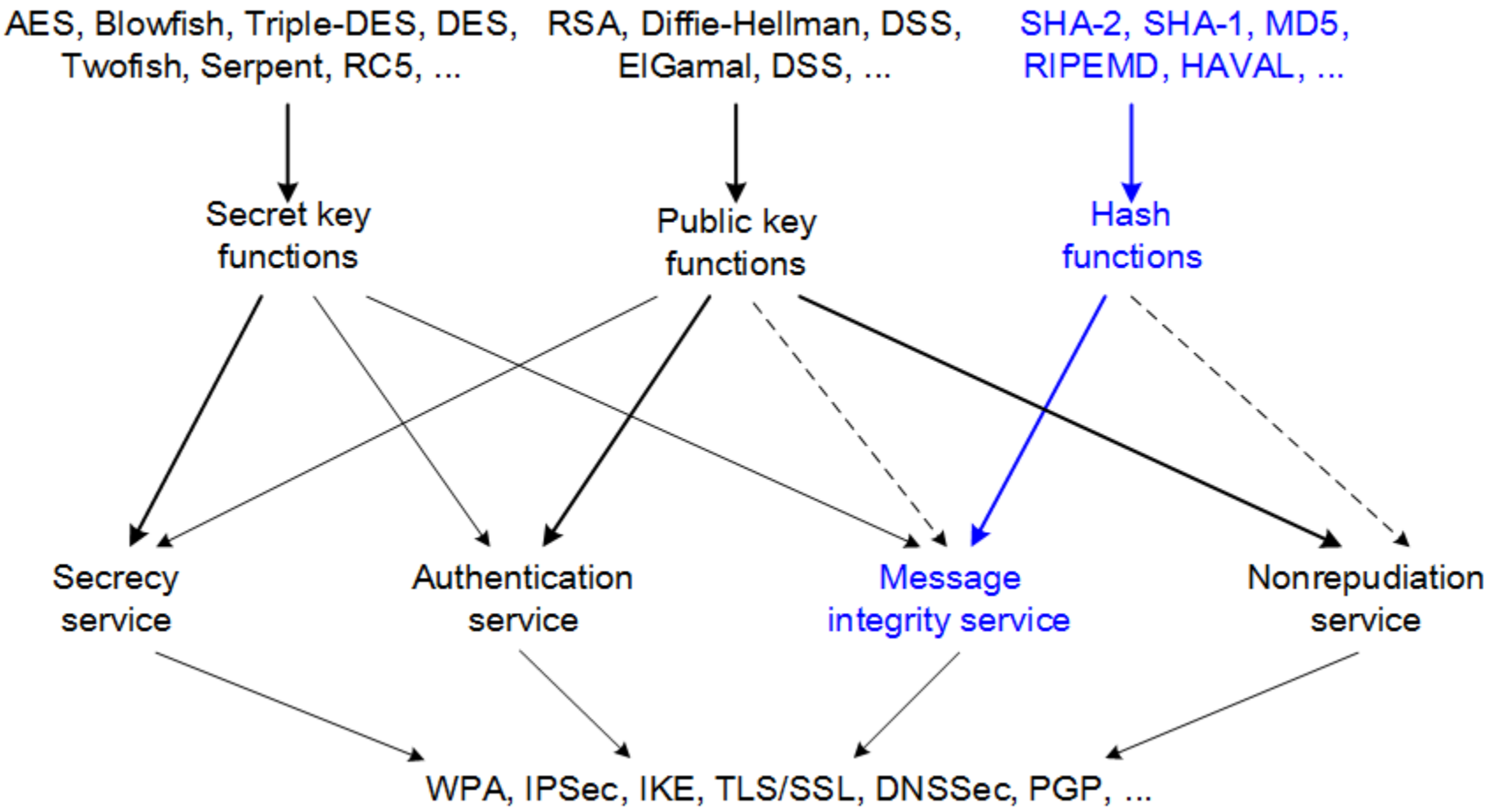
# Cryptographic Hash Functions

Rocky K. C. Chang, February 5, 2015

# This set of slides addresses



# Outline

- Cryptographic hash functions
  - Unkeyed and keyed hash functions
  - Security of cryptographic hash functions
  - Iterated hash functions
  - Two weaknesses
- Message authentication codes
  - What does an MAC do?
  - MAC security
  - HMAC
  - Using MAC properly

# Cryptographic hash functions

# Hash functions

- A hash function (or message digest function) takes an arbitrarily long string of bits and produces a fixed-sized result.
  - The hash result is also known as digest or fingerprint.
  - Cryptographic hash function vs. hashing used in data structures and algorithms.
  - Cryptographic hash function vs. error detection codes, such as checksum and CRC

# For examples,

- For a message m, compute x = h(m).
  - Assume that x is stored in a safe place, but m is not.
  - Whenever retrieving m, compute h(m).
    - If h(m)= x, one should be confident that m has not been altered.
- Alice and Bob share a secret key K, and use $h_K()$ to protect the integrity of their messages.
  - Assume that K is only known to Alice and Bob.
  - Alice (or Bob) computes $x = h_K(m)$ and sends (m, x) to Bob (or Alice).
  - At Bob's (or Alice) side, he computes $h_K(m)$.
    - If $h_K(m) = x$, (s)he should be confident that both m and x have not been altered.

# Many uses of cryptographic hash functions

- Message authentication (or message integrity) and digital signature
- Map a variable-sized value to a fixed-size value.
- Serve as a cryptographic pseudo-random generators to generate several keys from a single shared secret.
- Their one-way property isolates different parts of a system.

# A (keyed) hash family consists of

- **M**: a set of possible messages
- **X**: a finite set of possible message digests
- **K**: the key space, a finite set of possible keys
- For each $K \in \mathbf{K}$, there is a hash function $h_K \in \mathbf{H}$. Each $h_K$: **M** $\rightarrow$ **X**.
- Moreover,
  - Usually assume that $|\mathbf{M}| \geq 2|\mathbf{X}|$.
  - A pair (m, x) is *valid* under the key K if $h_K(m) = x$.
  - $|\mathbf{K}| = 1$ for unkeyed hash functions.

# Security of a cryptographic hash function

- The basic requirement for a cryptographic hash function is that
  - The only efficient way to produce a valid pair (m, x) is to first choose m, and then compute x = h(m).
- As a counter example, consider a message: $(m_1, m_2)$ with $h(m_1, m_2) = am_1 + bm_2 \bmod n$, where $m_1, m_2, a, b \in Z_n$, $n>1$.
  - Given $h(m_1, m_2)$ and $h(m'_1, m'_2)$, one can determine the value of h() for other messages.
  - For a message $(am_1+bm'_1, am_2+bm'_2)$, $h(am_1+bm'_1, am_2+bm'_2) = a\, h(m_1, m_2) + b\, h(m'_1, m'_2)$.
- Security of a cryptographic hash function can be evaluated based on the difficulty of solving three problems.

# Problem 1: The preimage problem

- *The preimage problem*:
  - Given a hash function h: **M** → **X** and an element x ∈ **X**,
  - Find m ∈ **M** such that h(m) = x.
- If the preimage problem can be solved, then (m, x) is a valid pair.
- A hash function for which the preimage problem cannot be efficiently solved is said to be *one-way* or *preimage resistant*.

# Problem 2: The second preimage problem

- *The second preimage problem*:
  - Given a hash function h: **M** → **X** and an element m ∈ **M**,
  - Find an m' ∈ **M** such that m' ≠ m and h(m') = h(m).
- If the 2nd preimage problem can be solved, then (m', h(m)) is a valid pair.
- A hash function for which the 2nd preimage problem cannot be <u>efficiently</u> solved is said to be *second preimage resistant*.

# Problem 3: The collision problem

- *The collision problem*:
  - Given a hash function h: **M** → **X**,
  - Find m, m' ∈ **M** such that m' ≠ m and h(m') = h(m).
- If (m, x) is a valid pair, and m, m' is a solution to the collision problem, then (m', x) is also a valid pair.
- A hash function for which the collision problem cannot be <u>efficiently</u> solved is said to be *collision resistant*.
- Which problem is the easiest to solve?

# Solving the preimage problem

- Consider the following algorithm to solve the preimage problem.
  1. Choose a subset $\mathbf{M_0} \subseteq \mathbf{M}$ and $|\mathbf{M_0}| = q$.
  2. For each $m \in \mathbf{M_0}$, if $h(m) = x$, return m.
  3. Return "unsuccessful."
- Pr[success] = 1 – Pr[all q attempts are unsuccessful].
- Assuming independent events, Pr[all q attempts are unsuccessful] = Pr[an attempt is unsuccessful]$^q$.
- Let $|\mathbf{X}|=B$ and Pr[an attempt is unsuccessful] = $1-1/B$.
- Therefore, Pr[success] = $1-(1-1/B)^q \approx q/B$ if q is small compared to B.

# Solving the 2nd preimage problem

- Consider the following algorithm to solve the 2nd preimage problem.
  1. Compute h(m).
  2. Choose a subset $\mathbf{M}_0 \subseteq \mathbf{M}\backslash\{m\}$ and $|\mathbf{M}_0| = q-1$.
  3. For each $m' \in \mathbf{M}_0$, if $h(m') = h(m)$, return $m'$.
  4. Return "unsuccessful."
- Pr[success] = $1-(1-1/B)^{q-1}$ .

# Solving the collision problem

- Consider the following algorithm to solve the collision problem.
  1. Choose a subset $\mathbf{M_0} \subseteq \mathbf{M}$ and $|\mathbf{M_0}| = q$.
  2. For each $m \in \mathbf{M_0}$, evaluate h(m).
  3. If h(m) = h(m') for some $m' \neq m$, return m', m.
  4. Else, return "unsuccessful."
- To conduct step 3, one can sort the values of h().

# Solving the collision problem

- Problem: what is the success probability of the algorithm to solve the collision problem given q attempts?
- Assume uniform probability and independence.
- Pr[unsuccessful] = Pr[all the q values of h() are different] = (B/B)((B–1)/B)((B–2)/B) … ((B–q+1)/B).
- Pr[successful] = 1–Pr[unsuccessful] = 1–(B/B)((B–1)/B)((B–2)/B) … ((B–q+1)/B).
- Pr[successful] ≈ $1 - e^{-q(q-1)/2B}$ for a sufficiently large B.

# The birthday attack

- Q: How many attempts are needed so that Pr[successful] ≥ p? (birthday problem if B = 365)
- After performing more approximation for Pr[successful] $\approx 1 - e^{-q(q-1)/2B}$ , we have
  - $q \approx (2B \ln(1/(1- \Pr[\text{successful}])))^{1/2}$.
- For p = 0.5, $q \approx 1.17\sqrt{B}$.
  - Hashing just over $\sqrt{B}$ random elements of **M** yields a collision probability of 0.5.
  - Different values of p will give different constant factors, but q is still proportional to $\sqrt{B}$.
  - For a n-bit hash function, a birthday attack (or square root attack) needs $2^{n/2}$ random hashes.
  - Answer for the birthday problem?
- Which problem is the easiest to solve?

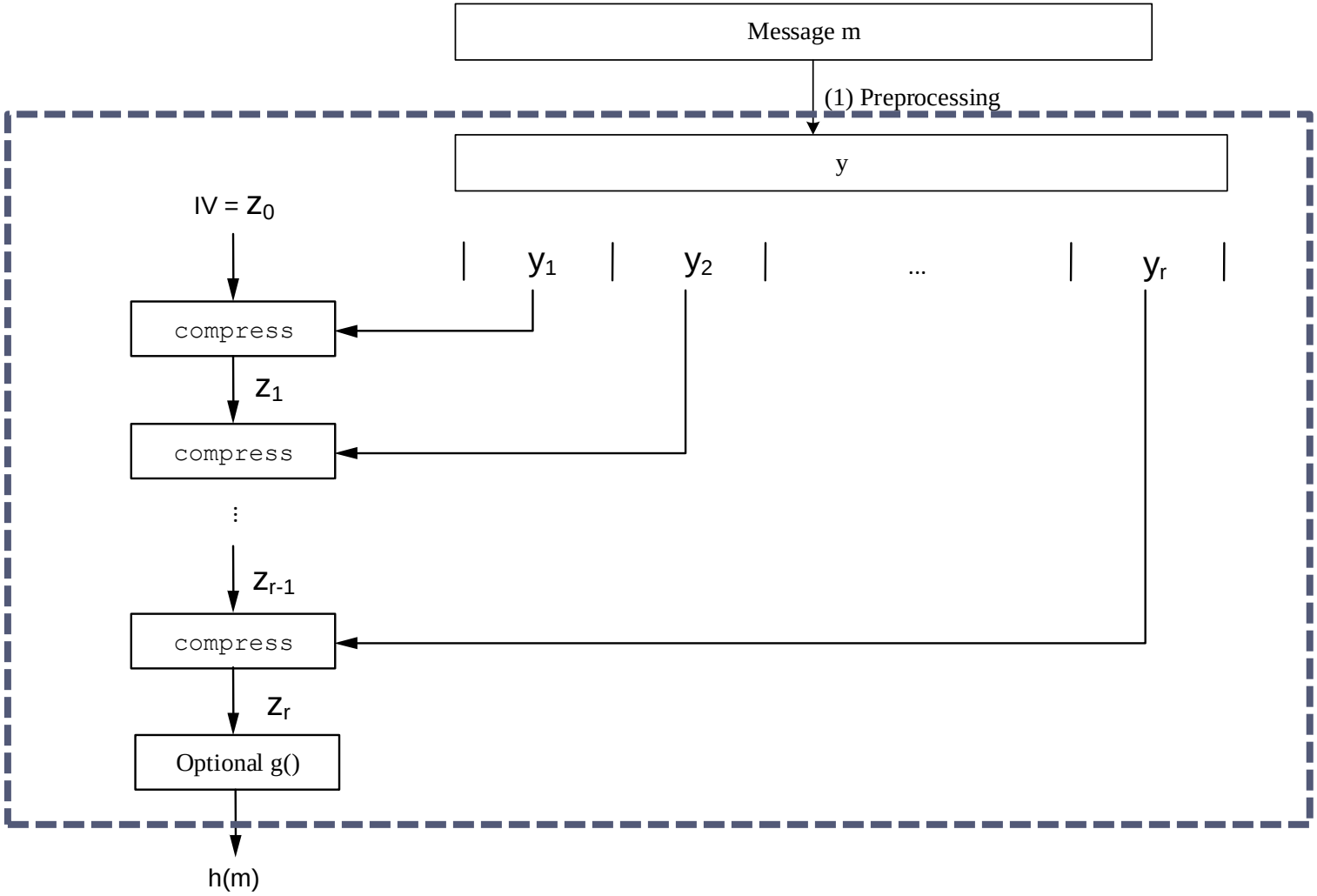
# Re-examining the 3 problems

- If we can solve the 2nd preimage problem, we can also solve the collision problem.
  - Randomly choose an m $\in$ **M**.
  - Use the solution to the 2nd preimage problem to find m'.
  - Return (m, m').
- If we can solve the preimage problem, we can also solve the collision problem.
  - Randomly choose an m $\in$ **M**.
  - Compute h(m).
  - Use the solution to the preimage problem to find m'.
  - Return (m, m').
- Collision resistant => 2nd preimage resistant and collision resistant => preimage resistant.

# Iterated hash functions

- Almost all hash functions put into practice are iterated hash functions.
  - h: **M** → **X**, where **X** = $\{0, 1\}^p$ (i.e., p-bit hash function).
- An iterated hash function h() usually consists of three main steps:
  - (1) Preprocessing
  - (2) Processing
  - (3) Output transformation
- Require a compression function for step (2):
  - `Compress` : $\{0,1\}^{n+t} \rightarrow \{0,1\}^n$, $t \geq 1$.

# Iterated hash functions



# (1) Preprocessing

- Given an input string m, where $|m| \geq n + t + 1$, construct a string y, such that $|y| \equiv 0 \pmod{t}$.
  - Let $y = y_1 \,||\, y_2 \,||\, \ldots \,||\, y_r$, where $|y_i| = t$, $i = 1, 2, \ldots, r$.
  - t is the *block size* and r is the number of blocks.
- This step must ensure that the mapping m→y is one-to-one.
  - Else, it is possible to find $m \neq m'$ so that $y = y'$.
  - Then $h(m) = h(m')$, i.e., h() would not be collision-resistant.
- Moreover, $|y| = rt \geq |m|$ because of the one-to-one requirement on the mapping m→y.
- A commonly used preprocessing step is to add padding: $y = m \,||\, pad(m)$.

# (2) Processing and (3) output transformation

- (2) Processing
  - Let IV be a public initial value of length n. Compute
    - $z_o \leftarrow IV$
    - $z_1 \leftarrow \texttt{compress}(z_o \,||\, y_1)$
    - $z_2 \leftarrow \texttt{compress}(z_1 \,||\, y_2)$
    - …
    - $z_r \leftarrow \texttt{compress}(z_{r-1} \,||\, y_r)$.
- (3) Optional output transformation
  - Let $g: \{0,1\}^n \rightarrow \{0,1\}^p$ be a public function. Without this transformation, we have n = p.

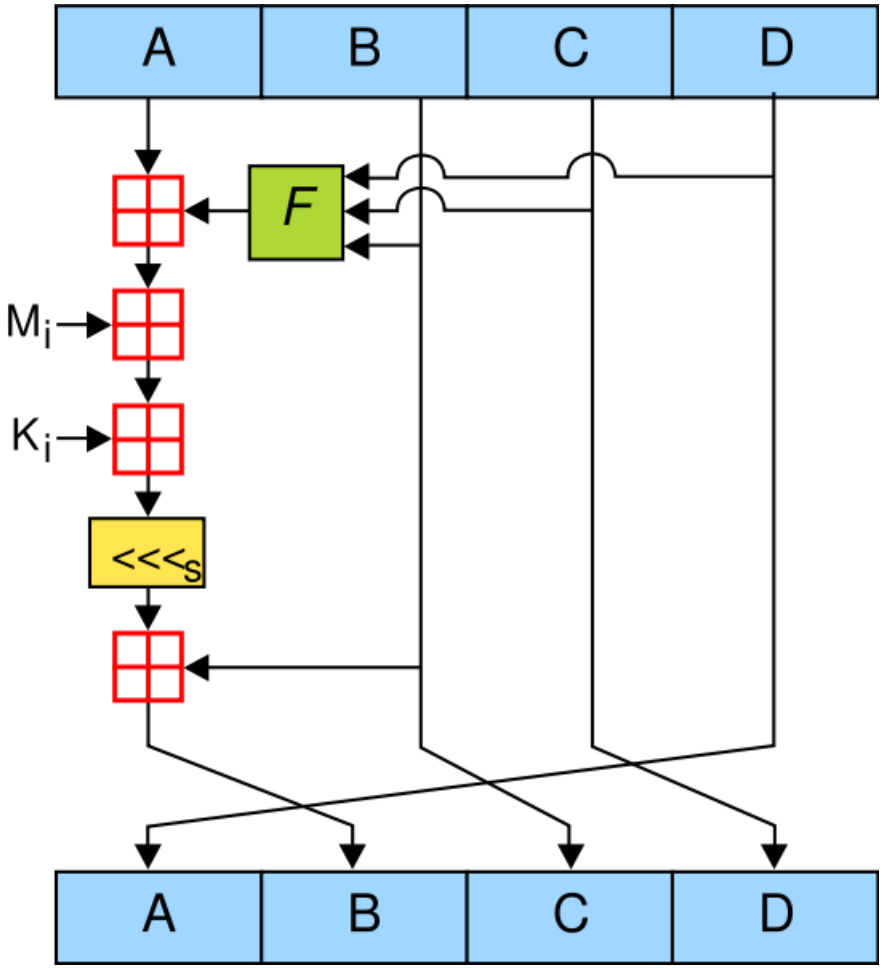
# Merkle–Damgård construction

- The construction is based on the iterated hash function construction with
  - The last block is padded with 0 and a binary string that encodes the length of the original message (Merkle–Damgård strengthening).
  - The `compress` function is collision-resistant.
  - Ralph Merkle and Ivan Damgård independently proved that the hash function is collision resistant if the `compress` function is collision-resistant.
- This construction was used in the design of many popular hash algorithms such as MD5 and SHA-1.

# Message Digest (MD5)

- Block size and output: t = 512 bits and p = 128 bits (4 x 32-bit)
- Padding
  - Padding is always performed.
  - The message is extended to just 64 bits short of a multiple of 512 bits long.
  - The last 64 bits encodes the message length.
  - For the rest: a single "1" bit is appended to the message, and then "0" bits are appended.
- The `compress` function is made from an "encryption function" by the Davies-Meyer scheme.
  - MD5 makes four passes over each block of data.
  - Each passes involves 16 operations.
- The hash output is a concatenation of the 4 output words.

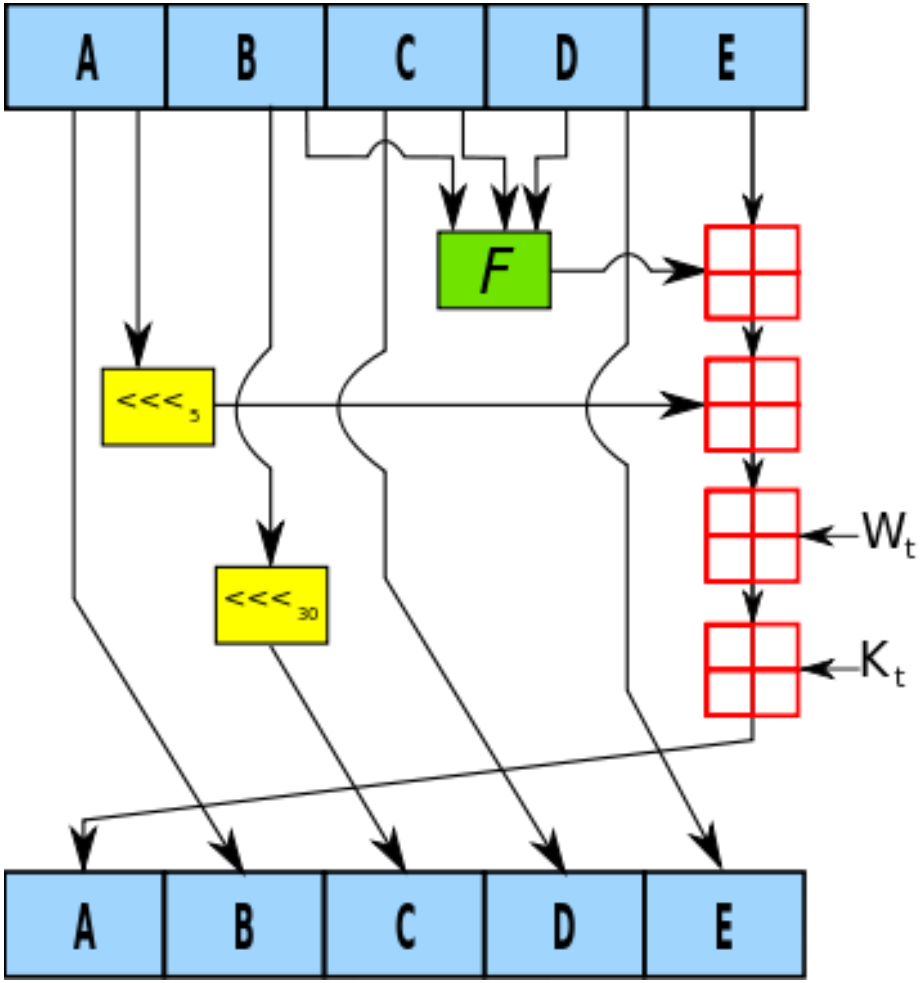
# A single operation in MD5 (wikipedia)



# Secure Hashing Algorithm (SHA-1)

- Block size and output: t = 512 bits and p = 160 bits (5 x 32-bit)
- Same padding as MD5
- The `compress` function is also made from an "encryption function" by the Davies-Meyer scheme.
  - SHA-1 makes five passes over each block of data.
  - Each rounds involves 20 operations.
- The hash output is a concatenation of the 5 output words.

# A single operation in SHA-1 (wikipedia)



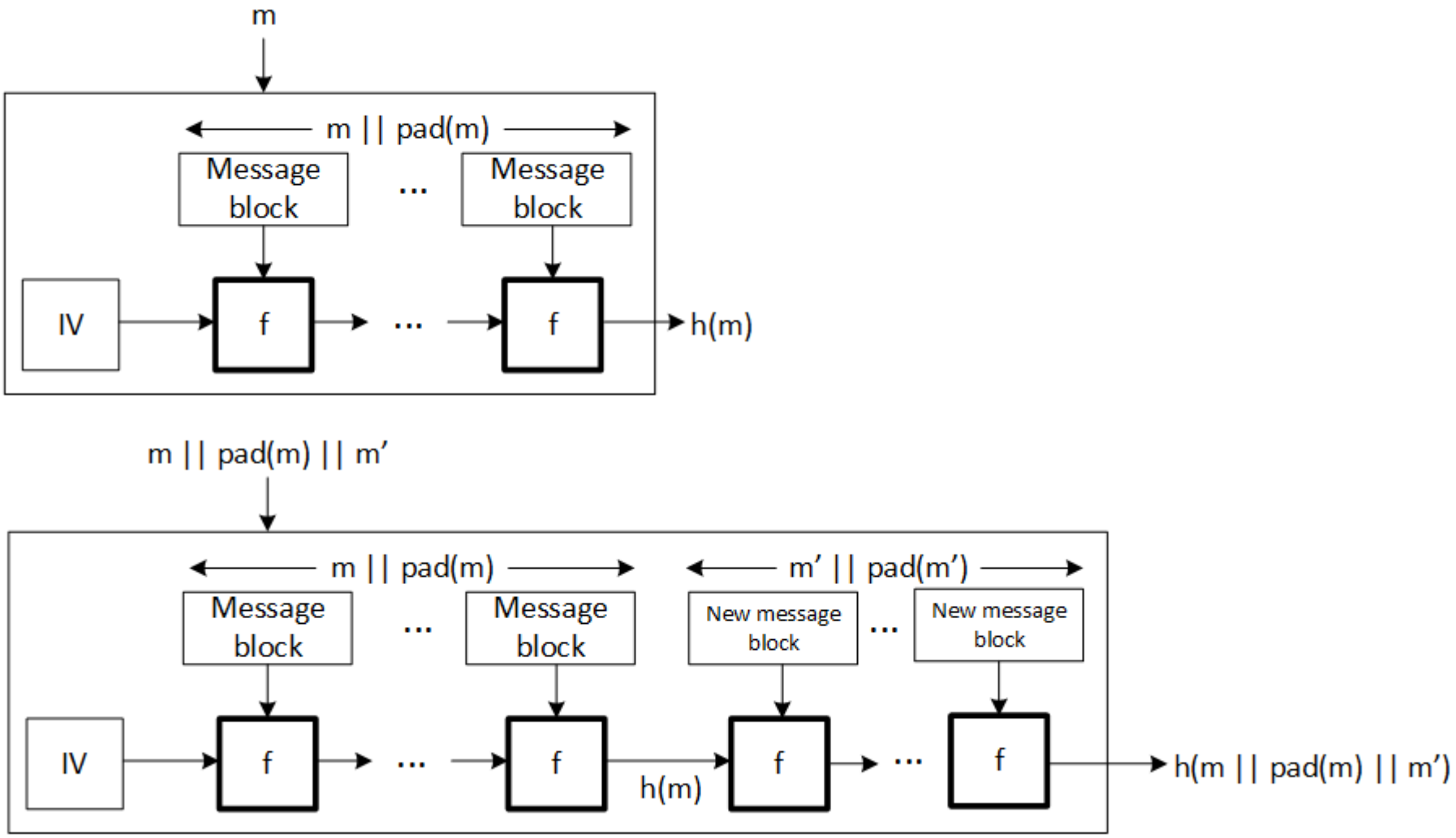
# Security of MD5 and SHA-1

- If the `compress` function is collision resistant, then the iterated hash function is also collision resistant.
- Security of MD5
  - The `Compress` function in MD5 is known to have collisions.
  - The 128-bit hash size is also insufficient.
- Security of SHA-1
  - SHA-1 was broken by a research team from Shandong University in 2005.
  - Collisions in the full SHA-1 in $2^{69}$ hash operations, much less than the brute-force attack of $2^{80}$ operations.
- SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)
- SHA-3, originally known as Keccak which was the winner of the NIST hash function competition in 2012.

# Weakness 1: length extensions

- Consider a message m which is hashed to a value h(m).
- Choose a new message that is m||pad(m)||m', where m' is an additional message.
- Therefore, h(m) is the intermediate hash value in the hash of the new message.
- Using h(m), m', and pad (m'), one can compute the new message's hash value.

# Weakness 1: length extensions (cont'd)



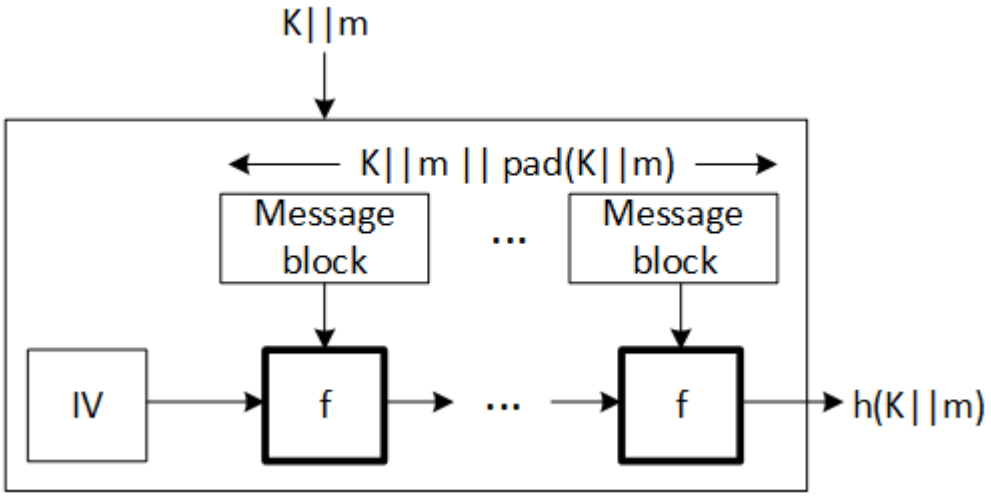
# What is the problem?

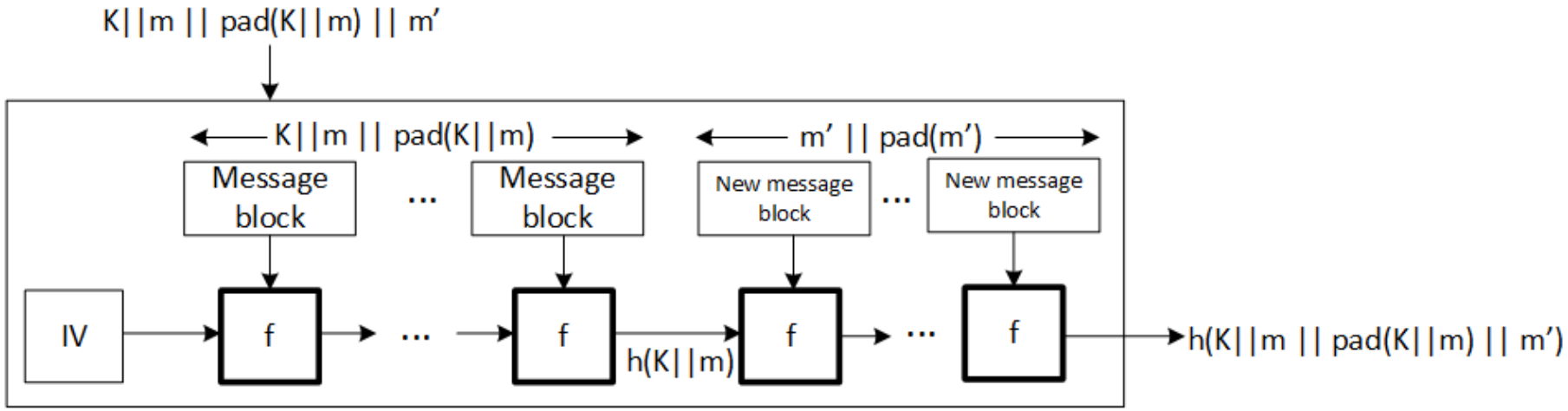
- The main problem is that there is no special processing at the end of the hash function computation.
- Consider that Alice sends a message to Bob and wants to authenticate it by sending h(K||m), where K is a secret shared by Alice and Bob.
- Now an attacker can append text to m, and update the hash value without knowing K.

# Workshop on the extension attack

- The attack
  - Attacker has the knowledge of h(K||m||pad(m)) and m.
  - Attacker will guess the length of the key K to compute pad(m).
  - With the correct key length, attacker can append arbitrary data to K||m with its paddings and obtain the correct hash of the appended message.

# The workshop (cont'd)



# Weakness 2: partial message collision

- Assume that mutual authentication is based on h(m||K), where
  - m is a random message and K is a secret key.
- How does an attacker obtain a correct h(m||K) without knowing K?
- First, the attacker has to find two strings m and m' that lead to a collision when hashed by h(), i.e., the birthday attack.
- After getting one side to authenticate m, i.e., receiving h(m||K), he can produce h(m'||K) for m'.
- Since h() is computed iteratively,
  - Once there is a collision (h(m) = h(m')) and
  - the rest of the hash inputs are the same (i.e., K),
  - the hash value stays the same too (i.e., h(m||K) = h(m'||K)).

# Message authentication codes

# Message authentication codes

- An MAC is a construction that prevents tampering (modify, replay) with messages.
  - Encryption does not prevent an attacker from manipulating messages.
- Like encryption, MACs use a secret key K known only to both Alice and Bob.
  - Alice sends a message m to Bob with a MAC value MAC(K,m).
  - Bob checks that the MAC value of the message is equal to MAC(K,m).

# Security of MAC

- Similar to hash functions, an ideal MAC(K,m) should be computationally indistinguishable from a random mapping.
- An attack on MAC is successful if
  - Given $(m_1,MAC(K,m_1))$, $(m_2,MAC(K,m_2))$, …, $(m_k,MAC(K,m_k))$,
  - An attacker is able to find a message m (not $m_1, m_2, \ldots, m_k$) together with its valid MAC(K,m).
- The success of the attack does not necessarily require a full knowledge of K.

# Generating the MAC

- There are 2 main approaches to generating MACs.
  - (CBC-MAC) Use of CBC and the MAC is the last block of the ciphertext.
  - (HMAC) Use keyed hash functions.
- The CBC-MAC is generally considered secure if the underlying cipher is secure.
  - A number of different collision attacks that limit its security level.
  - Avoid using the same key for encryption and authentication.

# Keyed hash functions

- Hash functions were not originally designed for message authentication.
- Authentication of what?
  - A message is sent from a certain source.
  - A message has not been modified after being sent.
  - A message is not an old message.
- The main problem is how to encode a shared secret into a hash function.

# A few possibilities

- The secret-prefix method: MAC(K,m) = h(K||m).
  - Subject to the length extension attack
- The secret-suffix method: MAC(K,m) = h(m||K).
  - Subject to the partial message collision attack
- The secret-prefix-suffix method: MAC(K,m) = h(K||m||K).
  - A 128-bit key can be recovered using $2^{67}$ known text-MAC pairs.

# HMAC

- HMAC computes h(K ⊕ opad || h(K ⊕ ipad || m)).
  - opad and ipad are specified constants, and they should have a large Hamming distance from each other.
  - The message m is hashed only once and the output is hashed again with the key.
  - HMAC uses hash function as a black-box.
  - h() can be any of the iterative hash functions, such as MD5 and SHA-1.
- The main idea is to "key" the initial states for a hash function.
- HMAC was chosen as the mandatory-to-implement authentication transform for IPSec (RFC 2104).

# Using MAC properly

- What information should be authenticated?
  - Or, what part of a packet should be included in MAC(K,m)?
- The Horton Principle: Authenticate what is being meant, not what is being said.
  - An MAC only authenticates a string of bytes (what is being said), but
  - Not necessary the interpretation of the message (what is meant).

# For example,

- The authenticated message may include
  - A "message ID" that prevents replay attack,
  - The source and destination of the message,
  - Protocol field, etc.
- In another case, Alice may use MAC to authenticate m = a || b || c, where a, b, and c are some data fields.
  - Additional (authenticated) information may be sent to Bob on how to interpret these data fields, in terms of their lengths, for example.

# Summary

- Examined the problems connected to the security of a cryptographic hash function.
- The birthday attack is a major attack on hash functions.
- All the practical hash functions, such as MD5 and SHA-1, are based on iterated hash functions which can be subject to
  - Length extension attacks and
  - partial message collision attacks
- Message authentication is based on MAC computed on a message and a shared secret.
- The MAC's security can be compromised for some keyed hash functions.
- Authenticate what is being meant, not what is being said.

# Acknowledgments


- The notes are prepared mostly based on
  - D. Stinson, *Cryptography: Theory and Practice, Chapman & Hall/CRC, Second Edition, 2002.*
  - N. Ferguson and B. Schneier, *Practical Cryptography*, Wiley, 2003.