Message Authentication Codes and Cryptographic Hash Functions

- Readings
  - Sections 2.6, 4.3, 5.1, 5.2, 5.4, 5.6, 5.7

Secret Key Cryptography: Insecure Channels and Media

- Confidentiality
  - Using a secret key cipher such as DES/CBC, we can assure that messages sent on a medium cannot be tapped by an eavesdropper
  - Distribute a secret key between two parties, then use encryption on the sender's side and decryption on the receiver's side using any of the block or stream ciphers
  - Can use the same technique for storing information on a disk: encrypt the information and decrypt when needed
  - Check the tool GnuPG: http://www.gnupg.org/
- Integrity?

Checksums and CRCs

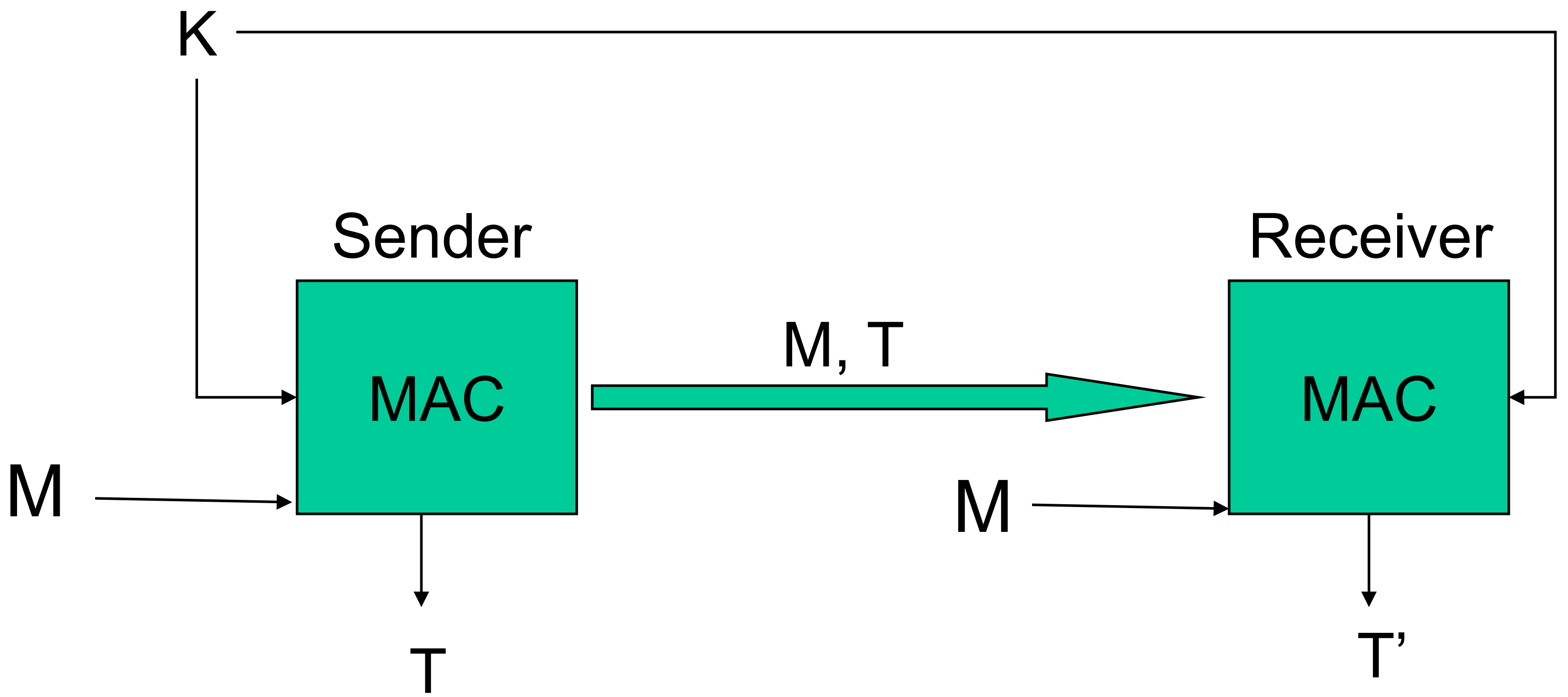
- Used to provide integrity checks against ***random faults***.
- **Not** sufficient for **protection against *malicious* or *intentional* modification**.
  - Easy to make changes and re-compute the CRC to match.
- In the past, it was believed that the use of CRCs within encryption was sufficient to provide integrity. However, that is no longer considered adequate:
  - Example: The use of CRCs in the WEP protocol resulted in a serious vulnerability, allowing for powerful active attacks.

Message Authentication Code (MAC)

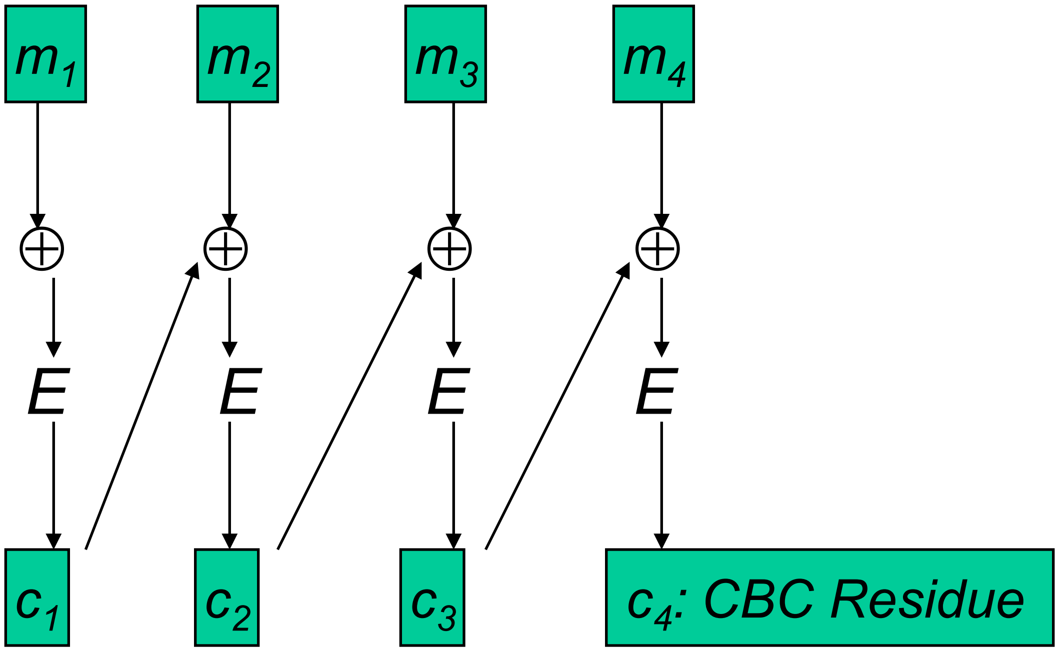
- A MAC is a cryptographic checksum that serves as an authenticator of the message
  - Generate a fixed length MAC (say 128 bits) from an arbitrary message
  - A “secret” key is used to generate the MAC
  - MAC should not be invertible

- The term message integrity code (MIC) is sometimes used and probably more accurate

Using MACs



MAC using Secret Key System



- Interested in integrity; not confidentiality
- Encrypt message; use last encrypted block as the MAC
- Send message + CBC residue
- $m_1 \,||\, m_2 \,||\, m_3 \,||\, m_4 \,||\, c_4$

Encryption + Integrity

- False solution:
  – Use a weak (non-cryptographic) checksum inside CBC: May prove to be completely insecure!
- Possible solutions
  1. Use two different keys in CBC mode (expensive).
  2. Use a different authentication mechanism, such as HMAC, which still requires processing the data twice, but less computationally costly.
  3. Use another encryption mode that provides both encryption and authentication (the future?)

Some care must be taken when combining encryption with MACs, in general

Order of Encryption/Authentication

- Encrypt then authenticate:
  - $E_{k'}(m) \,||\, MAC_{k''}(E_{k'}(m))$
- Generally secure, independent of the mode of encryption used
- Has the advantage to permit MAC verification before decryption (early compromise detection and avoidance of unnecessary cryptographic operations)

Authentication+Encryption

- Authenticate then encrypt:
  - $E_{k'}$ (m, $MAC_{k''}(m)$)
- Unsafe if a mode other than CBC is used.
- Provably secure with CBC.
- Does not permit verification before decryption.
- Authentication tag can be pre-computed, and remains associated with the original message after decryption.

Cryptographic Hash Functions

Hash Functions

- A hash function is a mathematical, efficiently computable function that has fixed size output:
  - $F : \{0, 1\}^N \rightarrow \{0,1\}^n$, where $N > n$
  - $F: \{0, 1\}^* \rightarrow \{0,1\}^n$
- In cryptography, the first type of hash function is often called a compression function, with the name hash function reserved for the unbounded domain type.
- Note: a hash function does not have a key and anyone can compute the same hash from the same message. However *keyed hashes* do use a key

Cryptographic Hash Functions

- Map a large space of values to a small space
- Hash values, also called message digests
- Cannot be easily invertible
- Examples:

  MD2, MD4, MD5 (128 bits), SHA-1(160 bits)

  SHA-2 (256/224 bits, 512/384 bits)

  What does it mean for a hash function to be broken?

Cryptographic Hash Functions

- The security of hash functions is defined empirically, if the following problems are found to be computationally infeasible:
  - One way:
    - Given y, find x such that h(x) = y
  - Second pre-image resistant:
    - Given x, find y$\neq$ x such that h(x) = h(y)
  - Collision-resistant:
    - Find y, x, with y$\neq$ x such that h(x) = h(y)

  - Can you prove that collision resistant (also called strong collision resistant) implies second pre-image resistant (also called weak collision resistant)?

One Way Function: what is computationally infeasible?

- Given y, find x such that h(x) = y
  - An inverse problem
  - If it is a cryptographic hash function with 128 bit digest, need to try about half of the messages to find a message that maps to y.
  - Need to try about 2^{127} messages
  - This should be computationally infeasible.

Second pre-image resistant - computational cost

- Given x, find y$\neq$ x such that h(x) = h(y)
  - Try messages to find some y that maps to the same value as x.
  - For a 128 bit digest, the expected number of messages to try is again about 2^{127} messages for a 0.5 probability of success

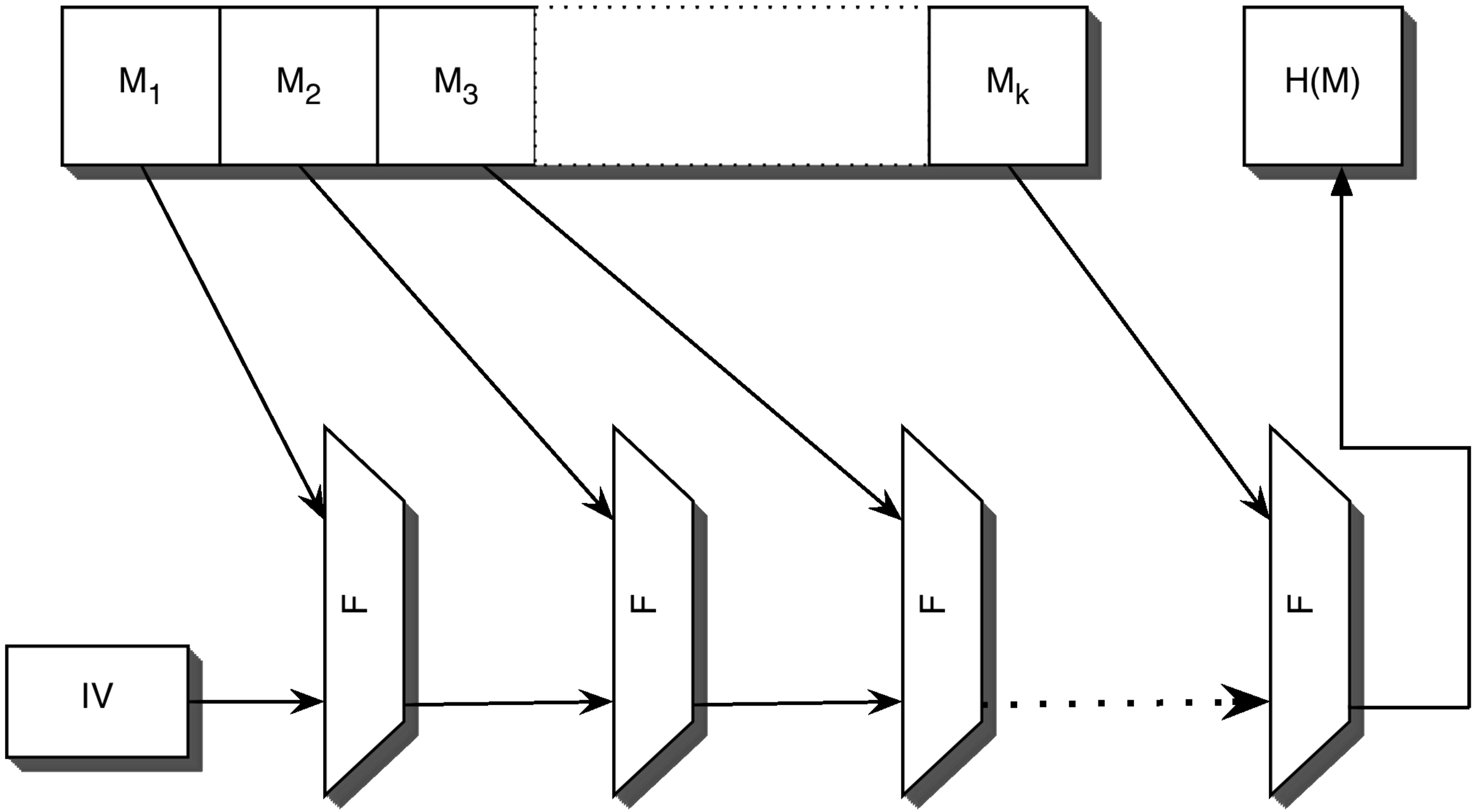
Collision Resistant - computational cost

- Find y, x, with y$\neq$ x such that h(x) = h(y)
  - For a cryptographic hash function, this requires solving the birthday problem, which is about 2^{64} messages for a 128 bit message digest
  - Note that collision resistant
    $\Rightarrow$ second pre-image resistant
  - Use ($A \Rightarrow B$ is the same as $\neg B \Rightarrow \neg A$)
  - Read more about the birthday problem
  - http://en.wikipedia.org/wiki/Birthday_problem

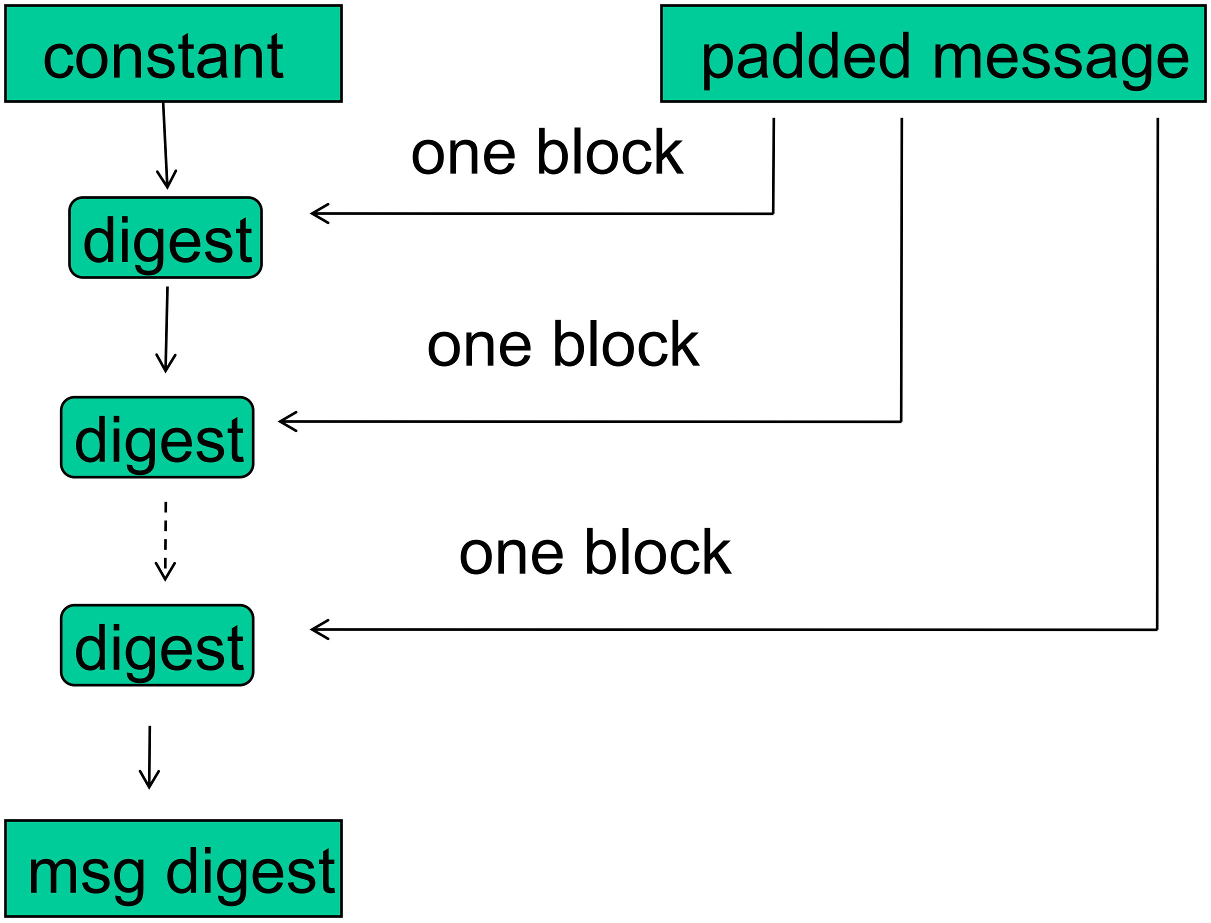
Constructing Hash Functions

- Since constructing secure hash functions is a difficult problem, the following approach has been taken in practice:
  - Construct a good compression function. Since the domain of compression functions are "small" they are easier to test for the desired properties.
- Use the MD construction (next) to turn a one-way, collision-resistant compression function into a hash function with similar properties.

Merkle-Damgard (MD)



A Somewhat Different View



MD5 (Message Digest 5)

- Invented by Ron Rivest
  - Considered broken and unsuitable for further use.

- Produce 128 bit message digest
- Assumes 32-bit **words**
- Let M be message to hash
- Pad M so length is 448 (mod 512)
  - Single “1” bit followed by “0” bits
  - At least one bit of padding, at most 512
  - Length before padding (64 bits) is appended
  - After padding message is a multiple of the 512-bit **block** size

MD5

- For 32-bit words A,B,C, define

  $F(A,B,C) = (A \wedge B) \vee (\neg A \wedge C)$ selection function

  $G(A,B,C) = (A \wedge C) \vee (B \wedge \neg C)$ selection function

  $H(A,B,C) = A \oplus B \oplus C$

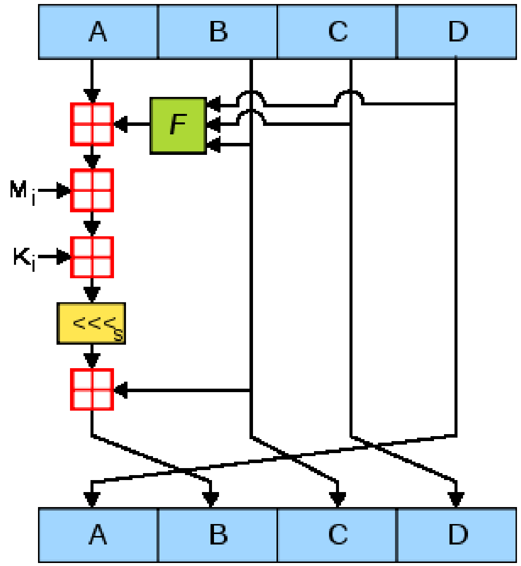
  $I(A,B,C) = B \oplus (A \vee \neg C)$
- Where $\wedge$, $\vee$, $\neg$, $\oplus$ are AND, OR, NOT, XOR, respectively

MD5

- Start out with 128 bit constant
  - 4 words (A, B, C, D)
- Operates in rounds (for each message block)
  - 4 rounds, 64 steps
  - Each round manipulates state using message block
  - Unique additive constant each step
  - Each step adds result of previous step

  - Round 0: Steps 0 thru 15, uses F function
  - Round 1: Steps 16 thru 31, uses G function
  - Round 2: Steps 32 thru 47, uses H function
  - Round 3: Steps 48 thru 63, uses I function

MD5: One Step Illustration



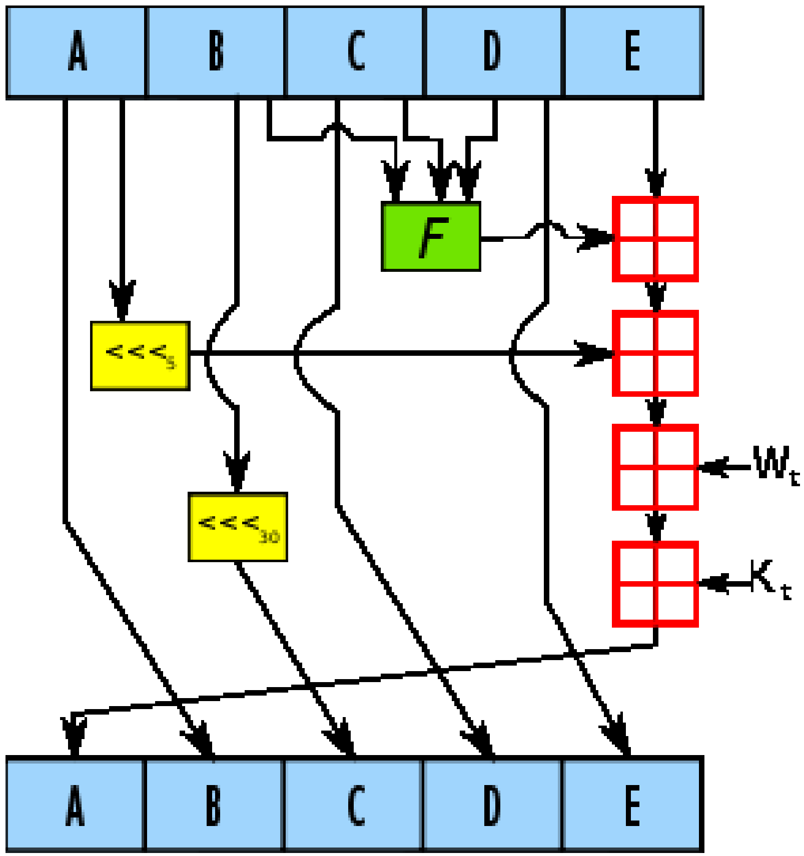
- K: unique constant in each step (32 bits)
- W: 32 bit block of input message
  - 32 * 16 = 512
- +: binary sum (carry out of high order discarded)
- <<<: left shift

SHA-1 (Secure Hash Algorithm)

- Operation is similar to MD5
  - more similar to MD4, which we did not discuss
  - In particular, the nonlinear functions used
- Produce 160 bit message digest
- 4 rounds, 20 steps each (for each message block)

- One of the series developed by NSA
  - Security attacks: 2^{51}, instead of 2^{80} brute force, as of 2008
  - SHA-3 development underway

SHA-1: One Step Illustration



Applications of Hash Functions

- System integrity protection:
- For password verification, eliminating the need to keep passwords
- As building blocks for **message authentication codes** (MACs) and **digital signature** algorithms.

- How to compute hash-based MAC?

Message Extension Attacks

- Since most hash functions are built using the Merkle-Damgard construction, they are vulnerable to length-extension attacks:
  - Given h(M) and length of M = len(M), but not M, an adversary can find h(M || M’) for chosen M’.
  - Secret key based MACs can be used to provide integrity of message transmission to prevent this attack
  - How about hash-based MAC? Can we add secret key into hash-based MAC?

- Also called Length Extension Attacks

Integrity Protection without Secret Key based MAC

- h(K || m) for integrity (know m and h())
  - Concatenate a secret key K with a message m and then use a cryptographic hash function
- h(m || K) (know m and h())
  - Concatenate the secret key K at the end
- Which (if either) is better?

H(K||M)?

- Hashes computed in blocks
- Recall $h(M_0,M_1) = F(h(M_0),M_1)$
- Let $M' = (M,X)$
  - Then $h(K,M') = F(h(K,M),X)$
  - Trudy can compute HMAC of M′ without K
  - Defeats the purpose of HMAC
- Length extension attack

How about H(M||K)?

- Is this better than h(K,M) ?

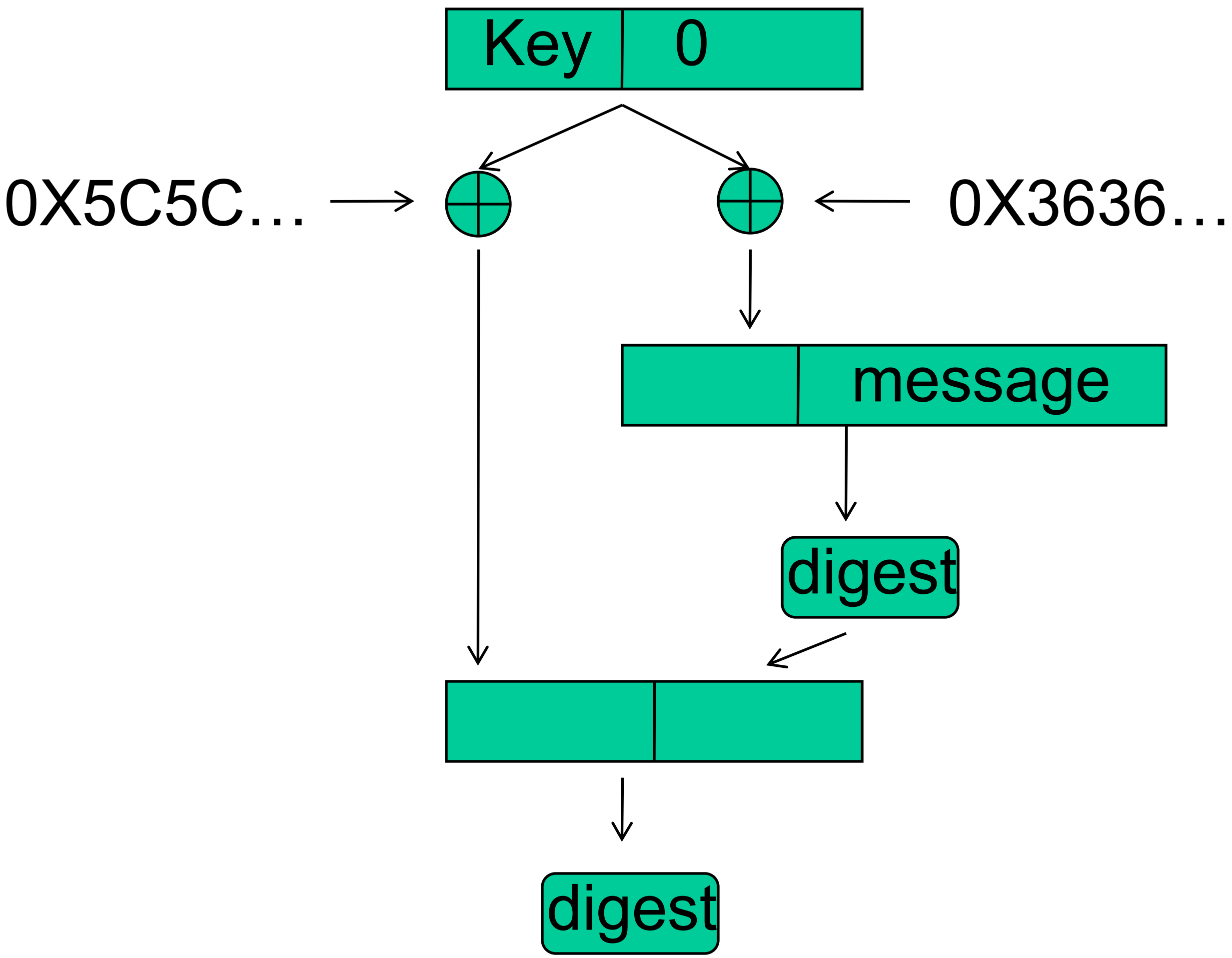
- If $h(M') = h(M)$ then

  $h(M,K) = F(h(M),K) = F(h(M'),K) = h(M',K)$
- In this case, Trudy can compute HMAC without knowing the key K
  - But collision must be known
  - Better than h(K,M), but we can do better

Building MACs from Hash Functions

- HMAC is a MAC from hash functions.
  - Hash-based message authentication code
- Let h be the hash function. Assume
- L = block length of compression function input
- Let K be the key, K' be the key padded with 0's to length L.
- HMAC(K, M) = $h(K' \oplus opad \,||\, h(K' \oplus ipad || M))$
- *ipad* = 0x363636...3636
- *opad* = 0x5c5c5c...5c5c (both of length L)

HMAC



- Why is it resilient to length extension attacks?

Reading Assignment

- Paper 7