# Cryptographic Hash Functions

## CSS322: Security and Cryptography

Sirindhorn International Institute of Technology
Thammasat University

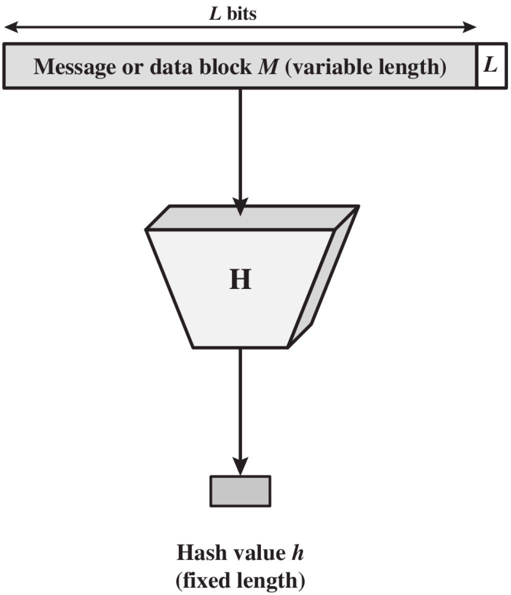
Prepared by Steven Gordon on 31 October 2012
CSS322Y12S2L08, Steve/Courses/2012/s2/css322/lectures/hash.tex, r2531

# Contents

# Hash Functions

- Hash function $\mathrm{H}$: variable-length block of data $M$ input; fixed-size hash value $h = \mathrm{H}(M)$ output
- Applying $\mathrm{H}$ to large set of inputs should produce evenly distributed and random looking outputs
- Cryptograhpic hash function: computationally infeasible to find:
  1. $M$ that maps to known $h$ (one-way property)
  2. $M_1$ and $M_2$ that produce same $h$ (collision-free property)
- Used to determine whether or not data has changed
- Examples: message authentication, digitial signatures, one-way password file, intrusion/virus detection, PRNG

# Cryptographic Hash Function

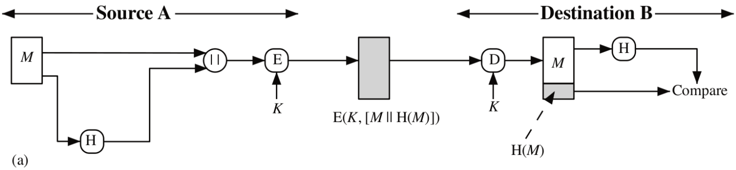


# Message Authentication

- Verify the integrity of a message
  - Ensure data received are exactly as sent
  - Assure identity of the sender is valid
- Hash function used to provide message authentication called message digest

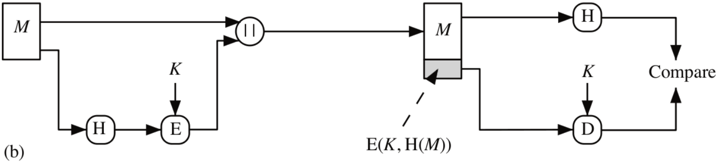
# Message Authentication Example (a)

- Encrypt the message and hash code using symmetric encryption



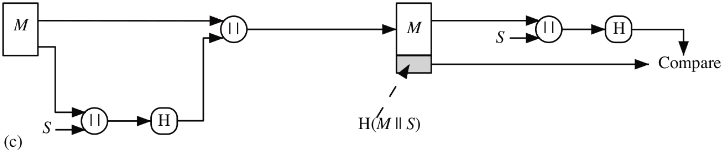
# Message Authentication Example (b)

- Encrypt only hash code
- Reduces computation overhead when confidentiality not required



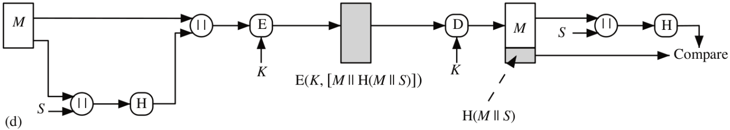
# Message Authentication Example (c)

- Shared secret $S$ is hashed
- No encryption needed



# Message Authentication Example (d)

- Shared secret combined with confidentiality

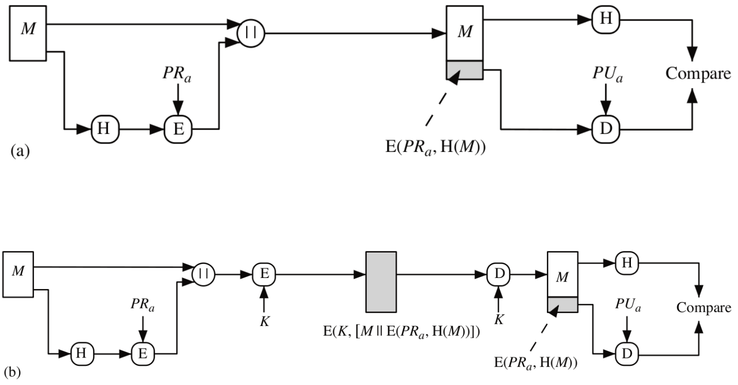


# Authentication and Encryption

- Sometimes desirable to avoid encryption when performing authentication
  - Encryption in software can be slow
  - Encryption in hardware has financial costs
  - Encryption hardware can be inefficient for small amounts of data
  - Encryption algorithms may be patented, increasing costs to use
- Message Authentication Codes (or keyed hash function)
  - Take secret key $K$ and message $M$ as input; produce hash (or MAC) as output
  - Combining hash function and encryption produces same result as MAC; but MAC algorithms can be more efficient than encryption algorithms
  - MAC covered in next topic

# Digital Signatures

- Hash value of message encrypted with user's private key
- Anyone with corresponding public key can verify integrity of message and author



# Contents

# Bit-by-Bit Exclusive OR

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

- $C_i$ is $i$th bit of hash code, $1 \leq i \leq n$
- $m$ is number of $n$-bit blocks in input
- $b_{ij}$ is $i$th bit in $j$th block
- Probability data error result in unchanged hash value: $2^{-n}$
- With structured data, effectiveness decreases

# Contents

# Preimages and Collisions

- For hash value $h = \mathrm{H}(x)$, $x$ is preimage of $h$
- $\mathrm{H}$ is a many-to-one mapping; $h$ has multiple preimages
- Collision occurs if $x \neq y$ and $\mathrm{H}(x) = \mathrm{H}(y)$
- Collisions are undesirable
- How many preimages for given hash value?
  - If $\mathrm{H}$ takes $b$-bit input block, $2^b$ possible messages
  - For $n$-bit hash code, where $b > n$, $2^n$ possible hash codes
  - On average, if uniformly distributed hash values, then each hash value has $2^{b-n}$ preimages

# Requirements of Cryptographic Hash Function

**Variable input size:** $\mathrm{H}$ can be applied to input block of any size

**Fixed output size:** $\mathrm{H}$ produces fixed length output

**Efficiency:** $\mathrm{H}(x)$ relatively easy to compute (practical implementations)

**Preimage resistant:** For any given $h$, computationally infeasible to find $y$ such that $\mathrm{H}(y) = h$ (*one-way property*)

**Second preimage resistant:** For any given $x$, computaionally infeasible to find $y \neq x$ with $\mathrm{H}(y) = \mathrm{H}(x)$ (*weak collision resistant*)

**Collision resistant:** Computationally infeasible to find any pair $(x, y)$ such that $\mathrm{H}(x) = \mathrm{H}(y)$ (*strong collision resistant*)

**Pseudorandomness:** Output of $\mathrm{H}$ meets standard tests for pseudorandomness

# Required Hash Properties for Different Applications

**Weak hash function:** Satisfies first 5 requirements (but not collision resistant)

**Strong hash function:** Also collision resistant

| | Preimage Resistant | Second Preimage Resistant | Collision Resistant |
|---|---|---|---|
| Hash + digital signature | yes | yes | yes* |
| Intrusion detection and virus detection | | yes | |
| Hash + symmetric encryption | | | |
| One-way password file | yes | | |
| MAC | yes | yes | yes* |

* Resistance required if attacker is able to mount a chosen message attack

# Brute Attacks on Hash Functions

## Preimage and Second Preimage Attack

- Find a $y$ that gives specific $h$; try all possible values of $y$
- With $m$-bit hash code, effort required proportional to $2^m$

## Collision Resistant Brute Attack

- Find any two messages that have same hash values
- Effort required is proportional to $2^{m/2}$
- Due to birthday paradox, easier than preimage attacks

## Practical Effort

- Crytpanalysis attacks possible in theory; complex
- Collision resistance desirable for general hash algorithms
- MD5 uses 128-bits: collision attacks possible ($2^{60}$)
- SHA uses longer codes; collison attacks infeasible

# Contents

# MD5

- Message Digest algorithm 5, developed by Ron Rivest in 1991
- Standardised by IETF in RFC 1321
- Generates 128-bit hash
- Was commonly used by applications, passwords, file integrity; no longer recommended
- Collision and other attacks possible; tools publicly available to attack MD5

# SHA

- Secure Hash Algorithm, developed by NIST
- Standardised by NIST in FIPS 180 in 1993
- Improvements over time: SHA-0, SHA-1, SHA-2, SHA-3
- SHA-1 (and SHA-0) are considered insecure; no longer recommended
- SHA-3 in development, competition run by NIST

| | SHA-1 | SHA-224 | SHA-256 | SHA-384 | SHA-512 |
|---|---|---|---|---|---|
| **Message Digest Size** | 160 | 224 | 256 | 384 | 512 |
| **Message Size** | $< 2^{64}$ | $< 2^{64}$ | $< 2^{64}$ | $< 2^{128}$ | $< 2^{128}$ |
| **Block Size** | 512 | 512 | 512 | 1024 | 1024 |
| **Word Size** | 32 | 32 | 32 | 64 | 64 |
| **Number of Steps** | 80 | 64 | 64 | 80 | 80 |