Cryptographic Hashes

CS 458: Information Security
Kevin Jin

Reading Material

- Text Chapters 2.2 and 21.1-2
- *Handbook of Applied Cryptography,* Menezes, van Oorschot, Vanstone
  - Chapter 9
  - http://www.cacr.math.uwaterloo.ca/hac/

What is Hash or Checksum?

- Mathematical function to generate a set of k bits from a set of n bits
  - $k \leq n$ except in unusual circumstances
- Example: ASCII parity bit
  - ASCII has 7 bits; 8th bit is "parity"
  - Even parity: even number of 1 bits
  - Odd parity: odd number of 1 bits

Example Use

- Bob receives “10111101” as bits.
  - Sender is using even parity; six 1 bits, so character was received correctly
    - Note: could be garbled, but 2 bits would need to have been changed to preserve parity
  - Sender is using odd parity; even number of 1 bits, so character was not received correctly

Another Example

- 8-bit Cyclic Redundancy Check (CRC)
  - XOR all bytes in the file/message
  - Good for detecting accidental errors
  - But easy for malicious user to "fix up" to match altered message
- For example, change the 4th bit in one of the bytes. How to "fix up"?
  - Fix up by flipping the 4th bit in the CRC
- Easy to find a M' that has the same CRC

Q: Uses of hash functions?

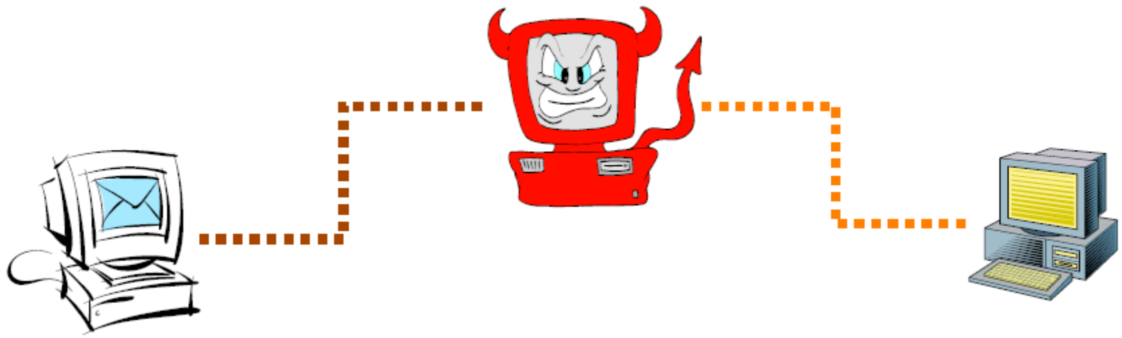
- Software integrity
  - E.g., tripwire
- Message authentication
- One-time Passwords
- Digital signature

Uses of hash functions

- Apache HTTP Server in .md5 file from web.
- Cisco MD5 for versions of IOS from Software Center on Cisco website.
- Darwin MD5 on web.
- Fedora Project SHA-1 on web and SHA1SUM file on ftp.
- FreeBSD on web and in CHECKSUM.MD5 and CHECKSUM.SHA256 files.
- GCC on ftp as md5.sum file.
- Gentoo as .md5 file on ftp.
- GNOME as MD5SUMS-for-gz and MD5SUMS-for-bz2 files on ftp.
- GnuPG SHA-1 on web.
- KDE on web and on ftp as MD5SUMS file.
- Knoppix in .md5 and .sha1 file.
- MySQL MD5 on web.
- OpenOffice.org MD5 on web.
- OpenSSH SHA-1 in release announcement.
- OpenSSL .md5 and .sha1 files linked to from web.
- Perl link to .md5 on web.
- PostgreSQL in a .md5 file.
- Python MD5 on web
- Ubuntu as MD5SUMS on ftp.
- X.org md5sums file on ftp.

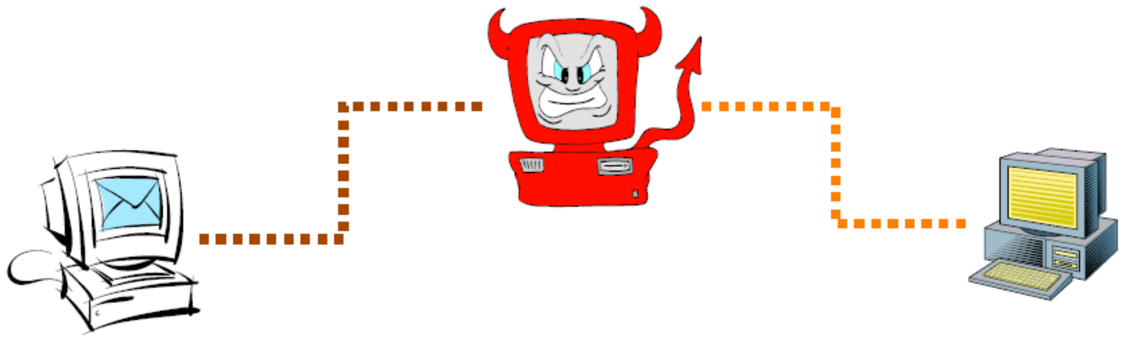
Source: http://microformats.org/wiki/hash-examples

Data Integrity and Source Authentication



- Integrity: detect unauthorized writing (i.e., modification of data)
- Encryption provides confidentiality (i.e., prevents unauthorized disclosure)
- Encryption alone does not provide integrity
  - One-time pad, ECB cut-and-paste, etc.

Data Integrity



When data integrity is more important than confidentiality?

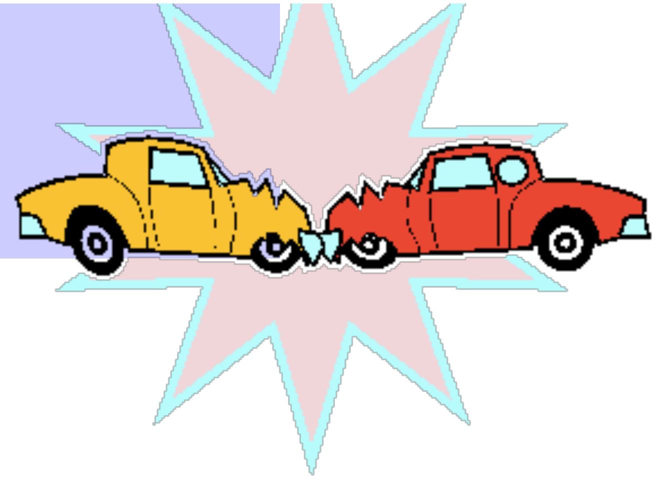
Example: Inter-bank fund transfers

Confidentiality may be nice, integrity is critical

Secure Hash functions

- Crytpo Hash or Checksum
  - Unencrypted **one–way** hash functions
  - Easy to compute hash
  - Hard to find message with a particular hash value
  - Use to verify integrity of publically available information
    - E.g., packets posted on mirror sites
- Message Authentication Code (MAC)
  - Hash to pass along with message
  - Such a hash must be accessed with key
    - Otherwise attacker could change MAC in transit

Collisions



- If x ≠ x′ and h(x) = h(x′), x and x′ are a ***collision***

Why collision could happen?

- Pigeonhole principle: if there are *n* containers for *n*+1 objects, then at least one container will have 2 objects in it.
- Application: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

Security Requirements for Cryptographic Hash Functions

Given a function $h: X \rightarrow Y$, then we say that h is:

- preimage resistant (one-way)
  if given $y \in Y$ it is computationally infeasible to find a value $x \in X$ s.t. $h(x) = y$
  e.g., computing x^3 vs cube root of x by hand
- 2nd preimage resistant (weak collision resistant)
  if **given** $x \in X$ it is computationally infeasible to find a value $x' \in X$, s.t. $x' \neq x$ and $h(x') = h(x)$
- collision resistant (strong collision resistant)
  if it is computationally infeasible to find two distinct values $x', x \in X$, s.t. $h(x') = h(x)$

Brute Force Attacks on Hash Functions

Attacking one-wayness

- Goal: given $h: X \rightarrow Y$, $y \in Y$, find x such that $h(x)=y$
- Algorithm:
  - pick a random value x in X, check if $h(x)=y$,
    if $h(x)=y$, returns x; otherwise iterate
  - after failing q iterations, return fail
- The average-case success probability (with replacement) is

$$\varepsilon = 1 - \left(1 - \frac{1}{|Y|}\right)^{q} \approx \frac{q}{|Y|}$$

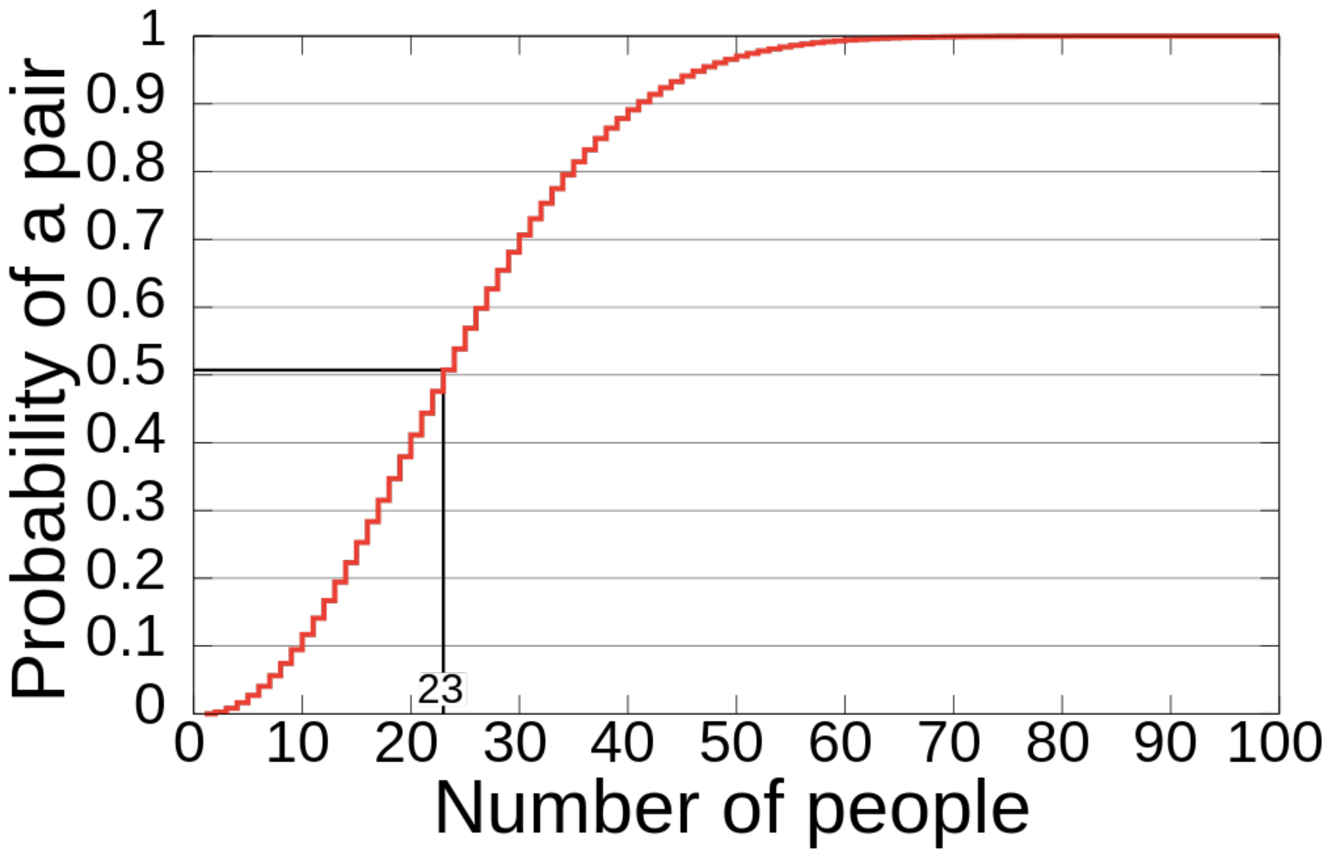
- Let $|Y|=2^m$, to get ε to be close to 0.5, $q \approx 2^{m-1}$

Birthday Paradox

- What is the probability that someone in the room has the same birthday as me?
- What is the probability that two people in the room have the same birthday?
  - $P(n) = 1 - (365! / (365^n * (365-n)!)$
  - Any 2 persons do not have the same bDay
    $= 364/365 * 363/365 * 362/365 \ldots 365-(n-1)/365$
    $= [365 * 364 * \ldots 365-(n-1)] / 365^n$
    $= [365! / (365-n)!] / 365^n$
  - $P(n) > ½$ for $n = 23$

  - Section 2.15 – Handbook of Applied Cryptography
  - http://en.wikipedia.org/wiki/Birthday_paradox

Birthday Paradox



Birthday Paradox

- In general, probability of a collision reaches 50% for M units when
  - $n = sqrt(M)$
- If hash has m bits, this means $M = 2^m$ possible hash values
  - $n = 2^{m/2}$ for 50% probability collision

Another View of Collisions

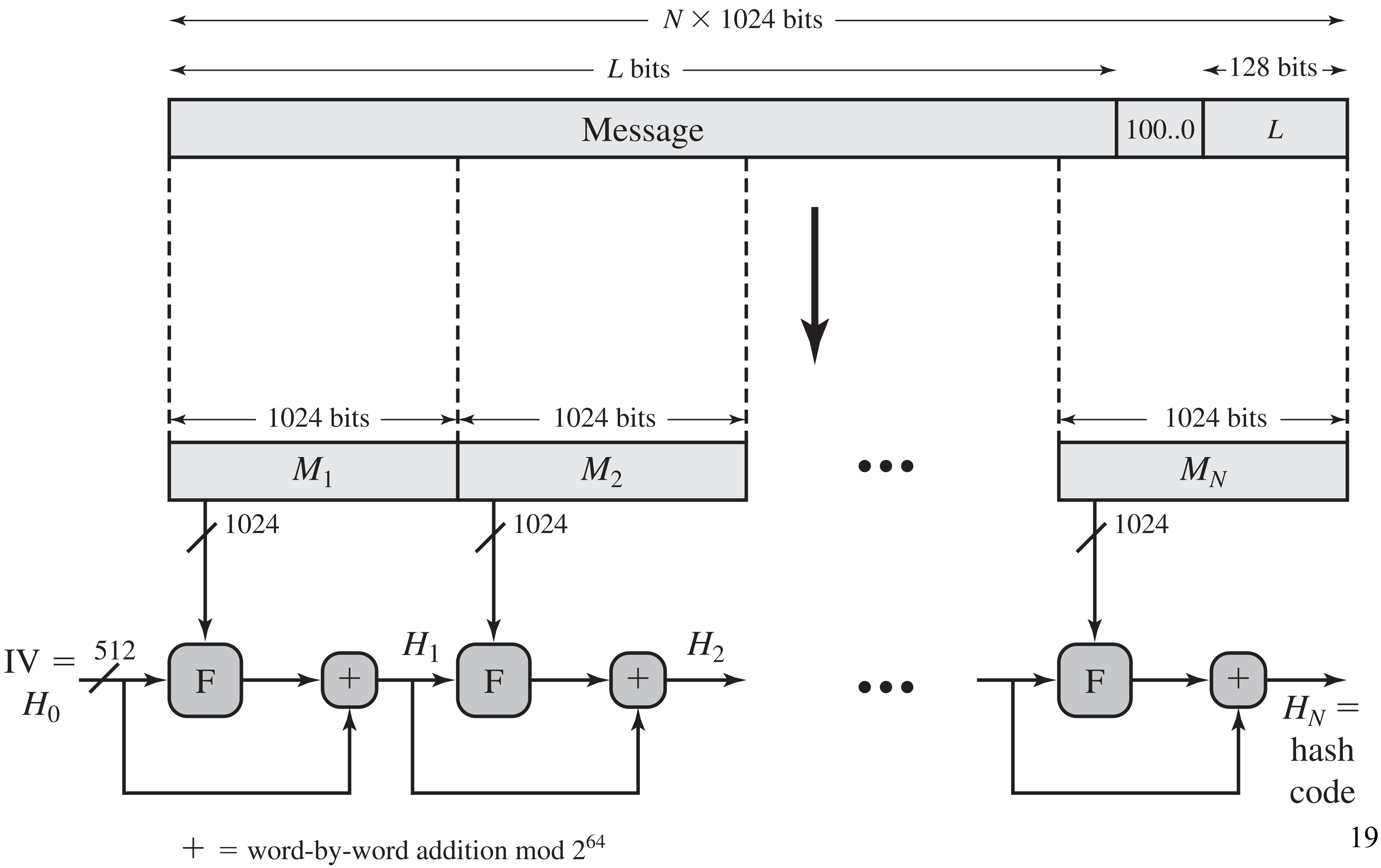
- **Birthday attack** works thus:
  – opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  – opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  – two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  – have user sign the valid message, then substitute the forgery which will have a valid signature
- Need to use larger MACs

MD5 and SHA

- Most widely used **keyless crypto hashes**
- Both are round based bit operations
  - Similar in spirit to AES and DES
  - Looking for avalanche effect to make output appear random
- MD5 is 128 bits and SHA-1 is 160 bit
- Problem with MD5?

  MD5 is only strong collision resistant to 2^{64} bits. Too small.

SHA


N × 1024 bits
L bits
128 bits
Message
100..0
L
1024 bits
1024 bits
1024 bits
M1
M2
MN
1024
1024
1024
IV = H0
512
F
+
H1
F
+
H2
F
+
HN = hash code
+ = word-by-word addition mod 2^64

Message Authentication Codes

- MAC is a crypto hash that is a proof of a message's integrity
  - Important that adversary cannot fix up MAC if he/she changes message
- MAC's rely on **keys** to ensure integrity
  - Similar to a hash augmented with a key

Hash vs. MAC

Hash

1. Alice->Bob: Hash(M)
   - Transmission must be *authentic* (integrity), need not be secret
2. Alice->Bob: M
   - Can use insecure channel
   - Integrity of M assured

MAC

1. Alice->Bob: K *(key)*
   - Transmission must be authentic *and confidential*
   - Only Alice and Bob know K
2. Alice->Bob: M, MAC_K(M)
   - Bob can verify integrity of M (Others cannot)
   - M does not have to be known ahead of time

Use Symmetric Ciphers for Keyed Hash

- Can use DES or AES in CBC mode
  - Last block is the hash
- DES with 64 bit block size is too small to be effective MAC

HMAC

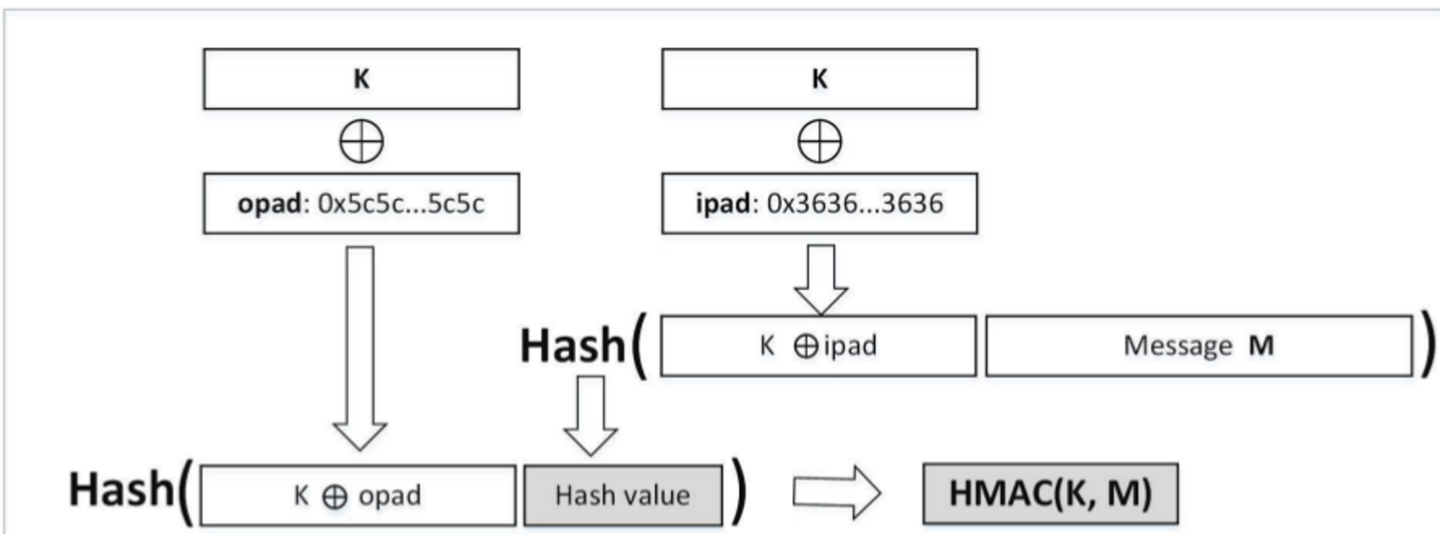
- Can compute a MAC of the message M with key K, using a “hashed MAC” or **HMAC**
- HMAC is a **keyed hash**
  – Why would we need a key?
- How to compute HMAC?
  – Two obvious choices: h(K,M) and h(M,K)
  – Which is better?

HMAC

- Should we compute HMAC as $\mathrm{h}(K, M)$?
- Hashes computed in blocks
  - $\mathrm{h}(B_1, B_2) = F(F(A, B_1), B_2)$ for some F and constant A
  - Then $\mathrm{h}(B_1, B_2) = F(\mathrm{h}(B_1), B_2)$
- Let $M' = (M, X)$
  - Then $\mathrm{h}(K, M') = F(\mathrm{h}(K, M), X)$
  - Attacker can compute HMAC of M' without K
- Is $\mathrm{h}(M, K)$ better?
  - Yes, but… if $\mathrm{h}(M') = \mathrm{h}(M)$ then we might have $\mathrm{h}(M, K) = F(\mathrm{h}(M), K) = F(\mathrm{h}(M'), K) = \mathrm{h}(M', K)$

The Right Way to HMAC

- Uses hash function H (compression function block size B) and a secret key K
- ipad = 0x36 (B times), opad = 0x5c (B times)
- Can be used with any one-way hash function



Example: HMAC-SHA512

- Apply HMAC to SHA512 to make a keyed MAC
- HMAC-SHA512(k, m) =

$$\mathrm{SHA512}(k' \oplus [01011100]^8 \,\|\, \mathrm{SHA512}(k' \oplus [00110110]^8 \,\|\, m))$$

$\oplus$ exclusive or, $\|$ concatenation

HMAC and Strong Collisions

- Birthday attacks don't make sense in HMAC scenario
  - Attacker would need to know K to generate candidate message/hash pairs
  - Thus HMAC-MD5 is still a reasonable option

Key Points

- Data integrity is important too
  - Sometimes more important than confidentiality
- Cryptohashes and Message Authentication Codes (MAC) both have their uses