Foundations of Network and Computer Security

John Black

Lecture #4
Sep 2nd 2004

CSCI 6268/TLEN 5831, Fall 2004

Announcements

- Please sign up for class mailing list

- Quiz #1 will be on Thursday, Sep 9th
  - About 30 mins
  - At end of class
  - Office hours day before and morning of
  - Covers all lecture materials and assigned readings

Blockcipher Review

- DES
  - Old, 64-bit blocksize, 56 bit keys
  - Feistel construction
  - Never broken except for exhaustive key search
- AES
  - New, 128-bit blocksize, 128-256 bit keys
  - Non-Feistel
  - Fast, elegant, so far so good

Aren't We Done?

- Blockciphers are only a start
  - They take n-bits to n-bits under a k-bit key
  - Oftentimes we want to *encrypt* a message and the message might be less than or greater than n bits!
  - We need a "mode of operation" which encrypts any $M \in \{0,1\}^*$
  - There are many, but we focus on three: ECB, CBC, CTR

ECB – Electronic Codebook

- This is the most natural way to encrypt
  - It's what we used with the Substitution Cipher
  - For blockcipher E under key K:
  - First, pad (if required) to ensure $M \in (\{0,1\}^n)^+$
  - Write $M = M_1 M_2 \ldots M_m$ where each M_i has size n-bits
  - Then just encipher each chunk:
    - $C_i = E_K(M_i)$ for all $1 \leq i \leq m$
  - Ciphertext is $C = C_1 C_2 \ldots C_m$

ECB (cont)

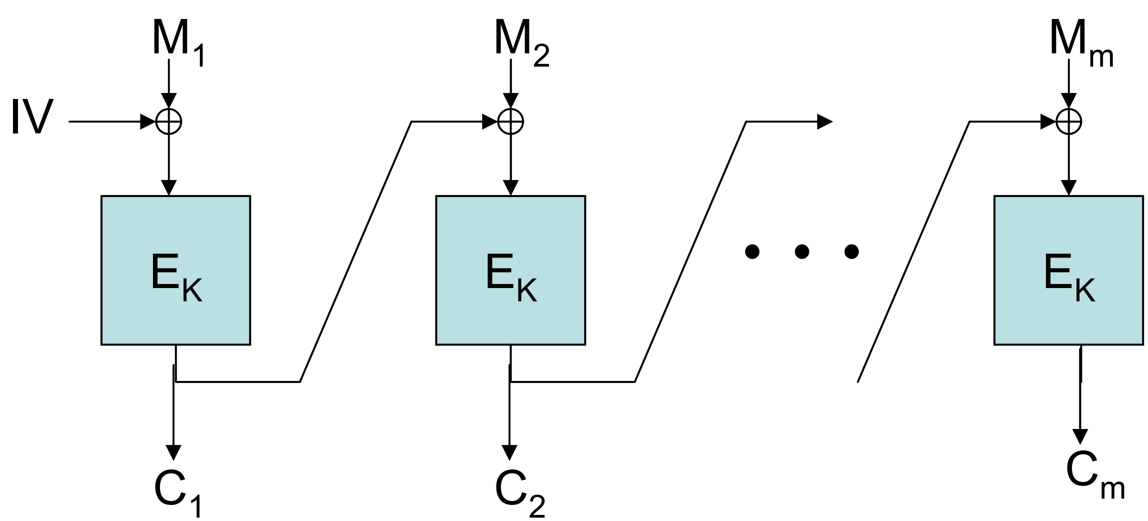
- What's bad about ECB?
  - Repeated plaintext blocks are evident in the ciphertext
    - Called "deterministic encryption" and considered bad
    - This was the feature of the Substitution Cipher that allowed us to do frequency analysis
    - Not as bad when n is large, but it's easy to fix, so why not fix it!
  - Encrypting the same M twice will yield the same C
    - Usually we'd like to avoid this as well

Goals of Encryption

- Cryptographers want to give up exactly two pieces of information when encrypting a message
  1) That M exists
  2) The approximate length of M
- The military sometimes does not even want to give up these two things!
  - Traffic analysis
- We definitely don’t want to make it obvious when a message repeats

CBC Mode Encryption

- Start with an n-bit "nonce" called the IV
  - Initialization Vector
  - Usually a counter or a random string
- Blockcipher E under key K, M broken into m blocks of n bits as usual
  - $C_0 = IV$
  - $C_i = E_K(M_i \oplus C_{i-1})$ for all $1 \leq i \leq m$



Features of CBC Mode

- Ciphertext is $C = C_0 C_1 \ldots C_m$
  - Ciphertext expansion of n-bits (because of C_0)
- Same block M_i, or same message M looks different when encrypted twice under the same key (with different IV's)
- No parallelism when encrypting
  - Need to know C_i before we can encipher M_{i+1}
  - Decryption *is* parallelizable however
- CBC mode is probably the most widely-used mode of operation for symmetric key encryption

Digression on the One-Time Pad

- Suppose Alice and Bob shared a 10,000 bit string K that was secret, uniformly random
  - Can Alice send Bob a 1KB message M with "perfect" security?
  - 1KB is 8,000 bits; let X be the first 8,000 bits of the shared string K
  - Alice sets $C = M \oplus X$, and sends C to Bob
  - Bob computes $C \oplus X$ and recovers M
    - Recall that $M \oplus X \oplus X = M$

Security of the One-Time Pad

- Consider any bit of M, m_i, and the corresponding bits of X and C, (x_i, c_i)
  - Then $c_i = m_i \oplus x_i$
  - Given that some adversary sees c_i go across a wire, what can he discern about the bit m_i?
    - Nothing! Since x_i is equally likely to be 0 or 1
  - So why not use the one-time pad all the time?
    - Shannon proved (1948) that for perfect security the key must be at least as long as the message
      - Impractical

One-Time Pad (cont)

- Still used for very-top-secret stuff
  - Purportedly used by Russians in WW II
- Note that it is very important that each bit of the pad be used at most one time!
  - The infamous “two time pad” is easily broken
    - Imagine $C = M \oplus X$, $C' = M' \oplus X$
    - Then $C \oplus C' = M \oplus X \oplus M' \oplus X = M \oplus M'$
    - Knowing the xor of the two messages is potentially *very* useful
    - n-time pad for large n is even worse (WEP does this)

Counter Mode – CTR

- Blockcipher E under key K, M broken into m blocks of n bits, as usual
- Nonce N is typically a counter, but not required

$$C_0 = N$$

$$C_i = E_K(N{+}{+}) \oplus M_i$$

- Ciphertext is $C = C_0\ C_1\ \ldots\ C_m$

CTR Mode

- Again, n bits of ciphertext expansion
- Non-deterministic encryption
- Fully parallelizable in both directions
- Not that widely used despite being known for a long time
  - People worry about counter overlap producing pad reuse

Why I Like Modes of Operation

- Modes are "provably secure"
  - Unlike blockciphers which are deemed "hopefully secure" after intense scrutiny by experts, modes can be proven secure like this:
    - Assume blockcipher E is secure (computationally indistinguishable from random, as we described)
    - Then the mode is secure in an analogous black-box experiment
      - The proof technique is done via a "reduction" much like you did in your NP-Completeness class
      - The argument goes like this: suppose we could break the mode with computational resources X, Y, Z. Then we could distinguish the blockcipher with resources X', Y', Z' where these resources aren't that much different from X, Y, and Z

Security Model

- Alice and Bob
  - Traditional names
  - Let's us abbreviate A and B
  - Adversary is the bad guy
    - This adversary is *passive*; sometimes called "eve"
  - Note also the absence of side-channels
    - Power consumption, timing, error messages, etc

Various Attack Models

- Known-Ciphertext Attack (KCA)
  - You only know the ciphertext
  - Requires you know something about the plaintext (eg, it’s English text, an MP3, C source code, etc)
  - This is the model for the Sunday cryptograms which use a substitution cipher
- Known-Plaintext Attack (KPA)
  - You have some number of plaintext-ciphertext pairs, but you cannot choose which plaintexts you would like to see
  - This was our model for exhaustive key search and the meet in the middle attack

Attack Models (cont)

- Chosen-Plaintext Attack (CPA)
  - You get to submit plaintexts of your choice to an encryption oracle (black box) and receive the ciphertexts in return
  - Models the ability to inject traffic into a channel
    - Send a piece of disinformation to an enemy and watch for its encryption
    - Send plaintext to a wireless WEP user and sniff the traffic as he receives it
  - This is the model we used for defining blockcipher security (computational indistinguishability)

Attack Models (cont)

- Chosen-Ciphertext Attack (CCA)
  - The strongest definition (gives you the most attacking power)
  - You get to submit plaintexts and ciphertexts to your oracles (black boxes)
  - Sometimes called a “lunchtime attack”
  - We haven’t used this one yet, but it’s a reasonable model for blockcipher security as well

So What about CBC, for example?

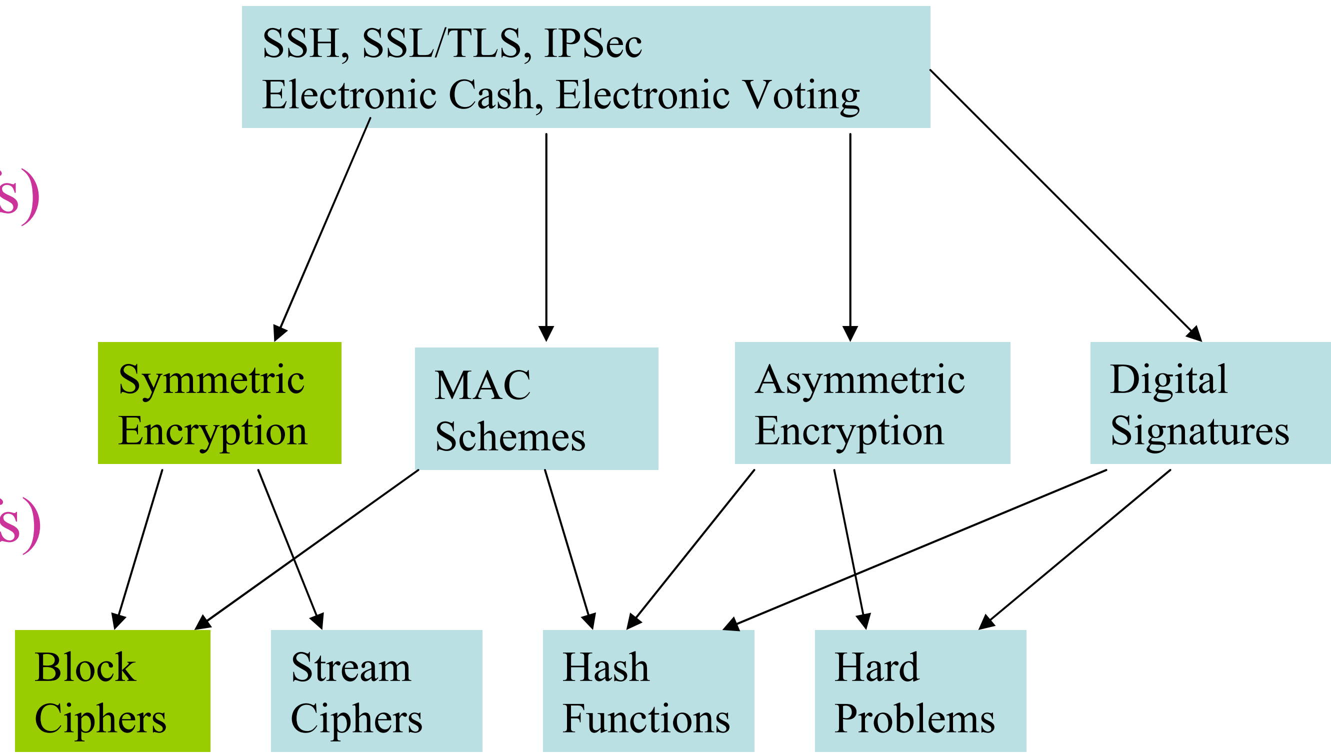
- CBC Mode encryption
  - It’s computationally indistinguishable under chosen plaintext attack
    - You can’t distinguish between the encryption of your query M and the encryption of a random string of the same length
  - In the lingo, “CBC is IND-CPA”
  - It’s not IND-CCA
    - You need to add authentication to get this

The Big (Partial) Picture

Second-Level Protocols
(Can do proofs)

First-Level Protocols
(Can do proofs)

Primitives



(No one knows how to prove security; make assumptions)

Symmetric Authentication: The Intuitive Model

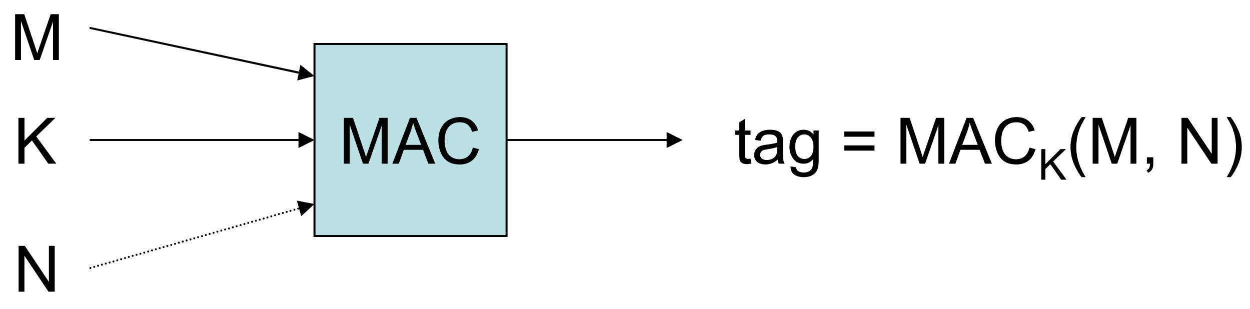
- Here's the intuition underlying the authentication model:
  - Alice and Bob have some shared, random string K
  - They wish to communicate over some insecure channel
  - An *active* adversary is able to eavesdrop and arbitrarily *insert* packets into the channel

Authentication: The Goal

- Alice and Bob's Goal:
  - Alice wishes to send packets to Bob in such a way that Bob can be certain (with overwhelming probability) that Alice was the true originator
- Adversary's Goal:
  - The adversary will listen to the traffic and then (after some time) attempt to impersonate Alice to Bob
  - If there is a significant probability that Bob will accept the forgery, the adversary has succeeded

The Solution: MACs

- The cryptographic solution to this problem is called a Message Authentication Code (MAC)
  - A MAC is an algorithm which accepts a message M, a key K, and possibly some state (like a nonce N), and outputs a short string called a “tag”



MACs (cont)

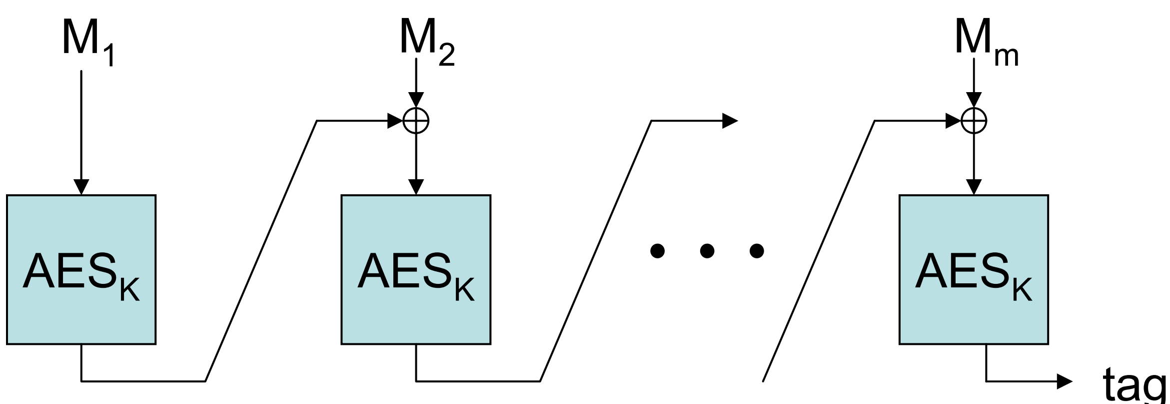
- Alice computes tag = $MAC_K(M, N)$ and sends Bob the message (M, N, tag)
- Bob receives (M', N', tag') and checks if $MAC_K(M', N')$ == tag'
  - If YES, he accepts M' as authentic
  - If NO, he rejects M' as an attempted forgery
- Note: We said nothing about privacy here! M might not be encrypted

Security for MACs

- The normal model is the ACMA model
  - Adaptive Chosen-Message Attack
- Adversary gets a black-box called an “oracle”
  - Oracle contains the MAC algorithm and the key K
  - Adversary submits messages of his choice and the oracle returns the MAC tag
  - After some “reasonable” number of queries, the adversary must “forge”
    - To forge, the adversary must produce a new message M^* along with a valid MAC tag for M^*
  - If no adversary can efficiently forge, we say the MAC is secure in the ACMA model

Building a MAC with a Blockcipher

- Let's use AES to build a MAC
  - A common method is the CBC MAC:
    - CBC MAC is stateless (no nonce N is used)
    - Proven security in the ACMA model provided messages are all of once fixed length
    - Resistance to forgery quadratic in the aggregate length of adversarial queries plus any insecurity of AES
    - Widely used: ANSI X9.19, FIPS 113, ISO 9797-1



CBC MAC notes

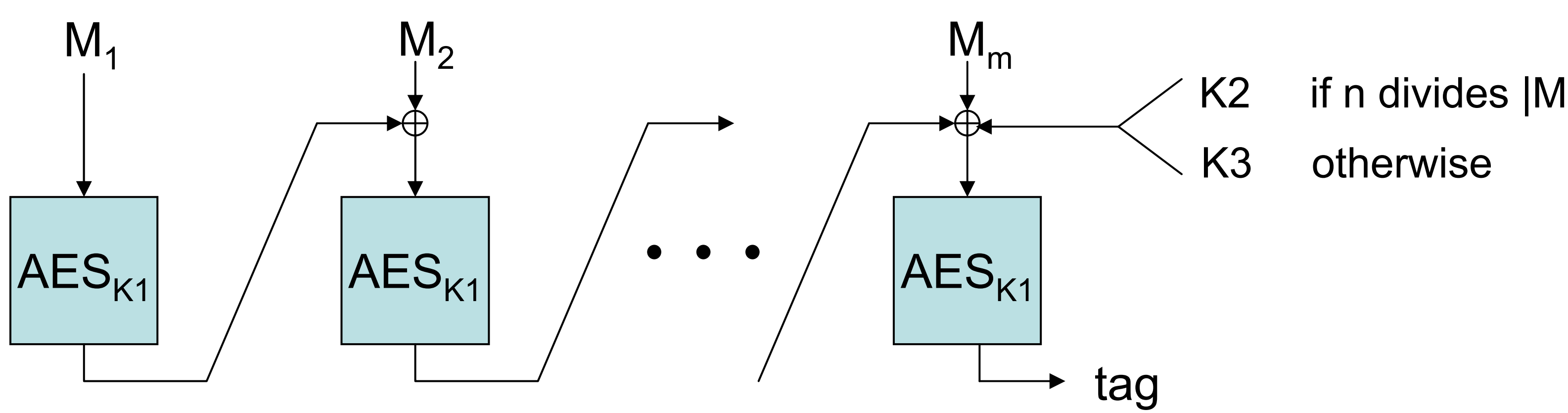
- Just like CBC mode encryption except:
  - No IV (or equivalently, IV is 0^n)
  - We output only the last value
- Not parallelizable
- Insecure if message lengths vary

Breaking CBC MAC

- If we allow msg lengths to vary, the MAC breaks
  - To "forge" we need to do some (reasonable) number of queries, then submit a new message and a valid tag
    - Ask $M_1 = 0^n$ we get $t = AES_K(0^n)$ back
    - We're done!
      - We announce that $M^* = 0^n \,||\, t$ has tag t as well
      - (Note that A || B denotes the concatenation of strings A and B)

Varying Message Lengths: XCBC

- There are several well-known ways to overcome this limitation of CBC MAC
- XCBC, is the most efficient one known, and is provably-secure (when the underlying block cipher is computationally indistinguishable from random)
  - Uses blockcipher key K1 and needs two additional n-bit keys K2 and K3 which are XORed in just before the last encipherment
- A proposed NIST standard (as “CMAC”)

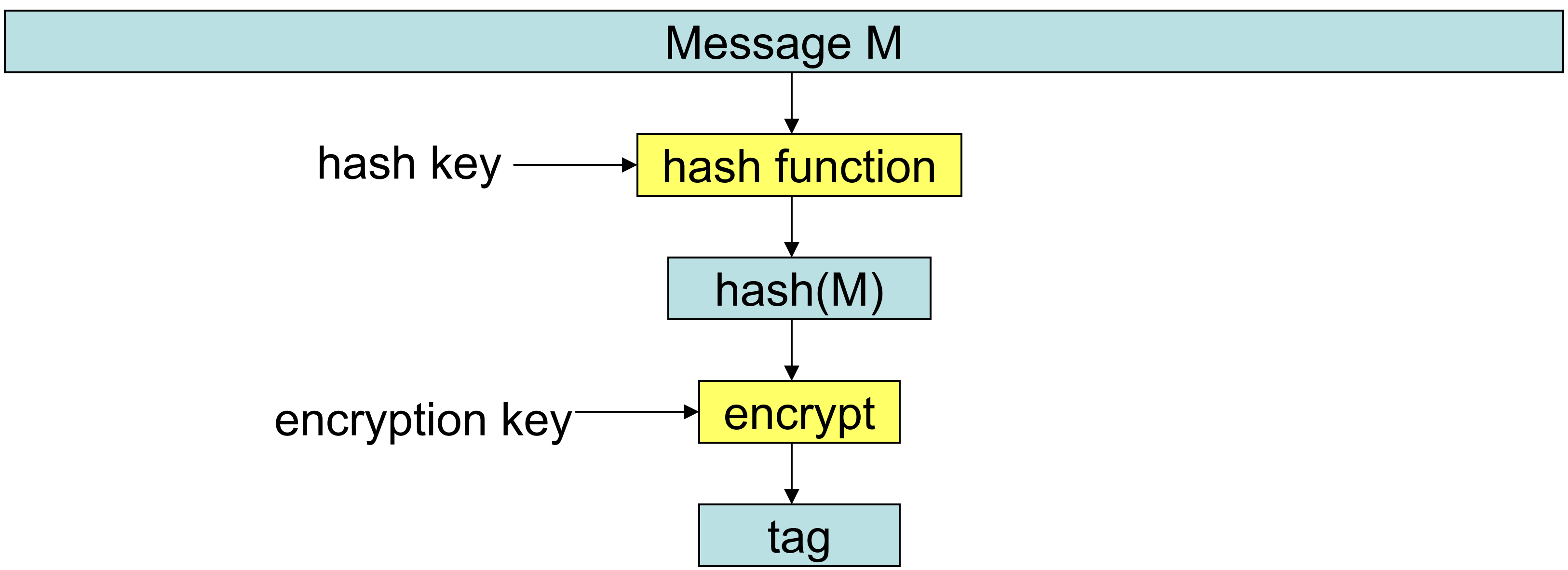


UMAC: MACing Faster

- In many contexts, cryptography needs to be as fast as possible
  - High-end routers process > 1Gbps
  - High-end web servers process > 1000 requests/sec
- But AES (a very fast block cipher) is already more than 15 cycles-per-byte on a PPro
  - Block ciphers are relatively expensive; it's possible to build faster MACs
- UMAC is roughly **ten times as fast** as current practice

UMAC follows the Wegman-Carter Paradigm

- Since AES is (relatively) slow, let's avoid using it unless we have to
  - Wegman-Carter MACs provide a way to process M first with a non-cryptographic hash function to reduce its size, and then encrypt the result



The Ubiquitous HMAC

- The most widely-used MAC (IPSec, SSL, many VPNs)
- Doesn't use a blockcipher or any universal hash family
  - Instead uses something called a "collision resistant hash function" H
    - Sometimes called "cryptographic hash functions"
    - Keyless object – more in a moment
    - $HMAC_K(M) = H(K \oplus opad \,||\, H(K \oplus ipad \,||\, M))$
    - opad is 0x36 repeated as needed
    - ipad is 0x5C repeated as needed

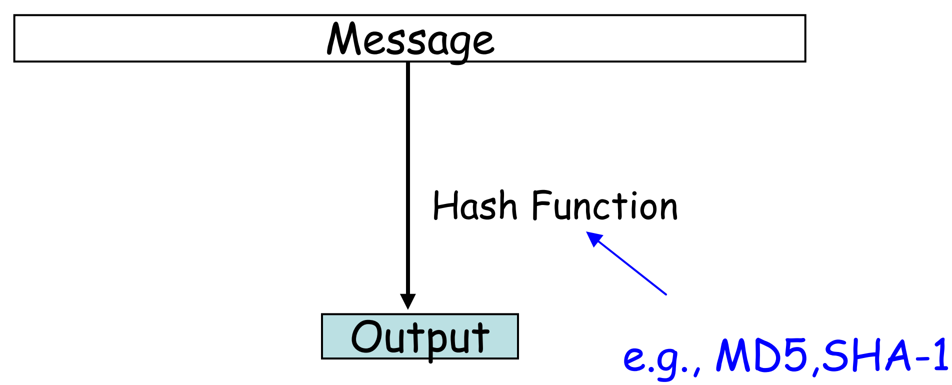
Notes on HMAC

- Fast
  - Faster than CBC MAC or XCBC
    - Because these crypto hash functions are fast
- Slow
  - Slower than UMAC and other universal-hash-family MACs
- Proven security
  - But these crypto hash functions have recently been attacked and may show further weaknesses soon

What are cryptographic hash functions?

- A cryptographic hash function takes a message from $\{0,1\}^*$ and produces a fixed size output
  - Output is called “hash” or “digest” or “fingerprint”
  - There is *no key*
  - The most well-known are MD5 and SHA-1 but there are other options
    - MD5 outputs 128 bits
    - SHA-1 outputs 160 bits

```
% md5
Hello There
^D
A82fadb196cba39eb884736dcca303a6
%
```



SHA-1

512 bits → M_1

M_1	M_2	…	M_m

for i = 1 **to** m **do**

$$W_t = \begin{cases} t\text{-th word of } M_i & 0 \le t \le 15 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) << 1 & 16 \le t \le 79 \end{cases}$$

$A \leftarrow H_0^{i-1};\quad B \leftarrow H_1^{i-1};\quad C \leftarrow H_2^{i-1};\quad D \leftarrow H_3^{i-1};\quad E \leftarrow H_4^{i-1}$

for t = 1 **to** 80 **do**

$T \leftarrow A << 5 + g_t(B, C, D) + E + K_t + W_t$

$E \leftarrow D;\quad D \leftarrow C;\quad C \leftarrow B >> 2;\quad B \leftarrow A;\quad A \leftarrow T$

end

$H_0^i \leftarrow A + H_0^{i-1};\quad H_1^i \leftarrow B + H_1^{i-1};\quad H_2^i \leftarrow C + H_2^{i-1};$

$H_3^i \leftarrow D + H_3^{i-1};\quad H_4^i \leftarrow E + H_4^{i-1}$

end

return $H_0^m\ H_1^m\ H_2^m\ H_3^m\ H_4^m$ ← 160 bits

Real-world applications

Hash functions are pervasive

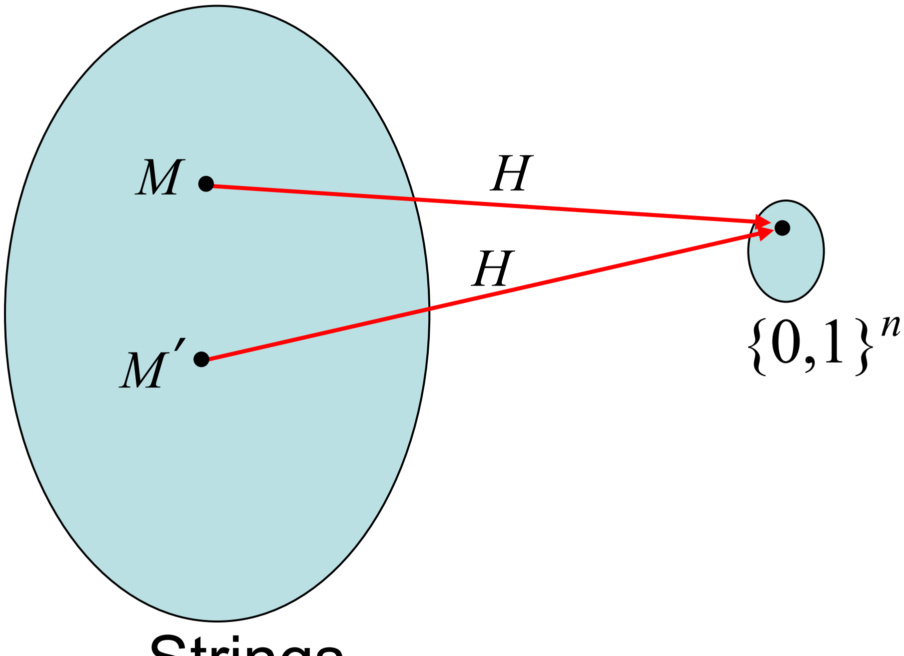
- Message authentication codes (HMAC)
- Digital signatures (hash-and-sign)
- File comparison (compare-by-hash, eg, RSYNC)
- Micropayment schemes
- Commitment protocols
- Timestamping
- Key exchange
- ...

A cryptographic property

(quite informal)

1. Collision resistance   given a hash function
it is hard to find **two colliding inputs**

BAD: $H(M) = M \bmod 701$



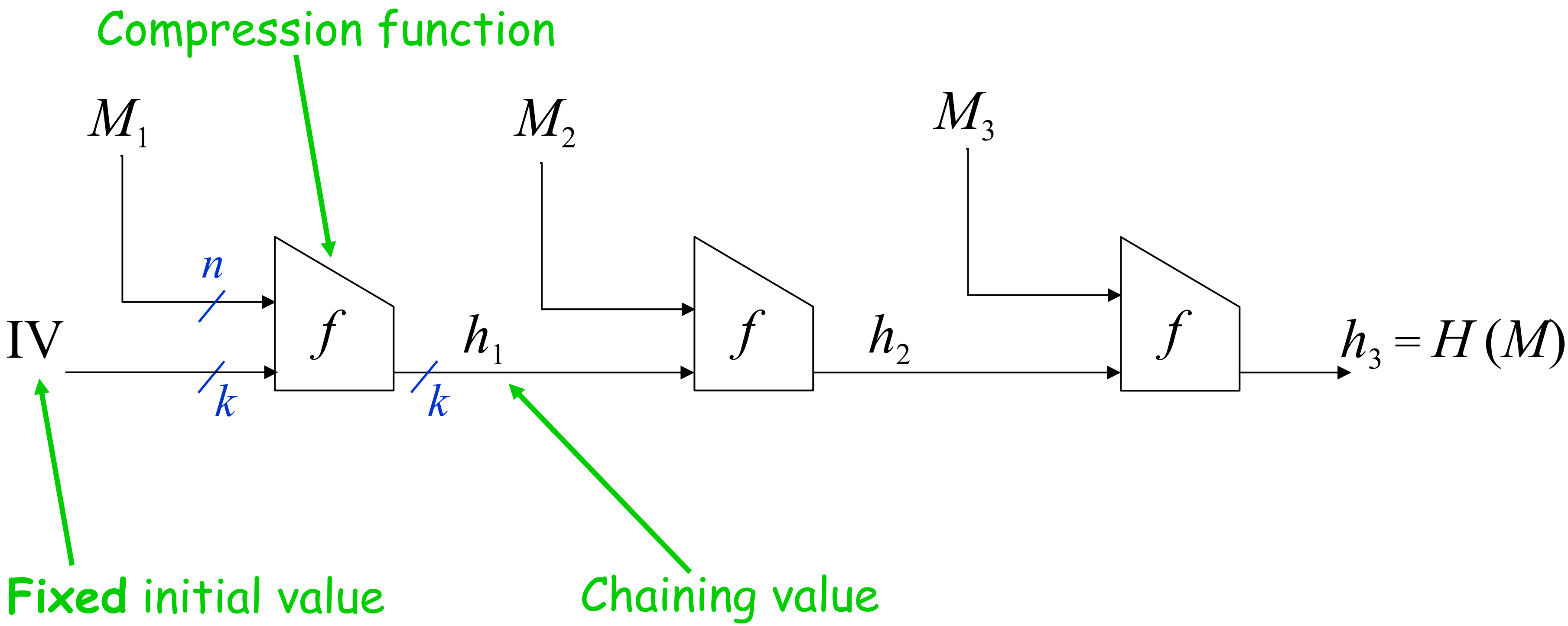
More cryptographic properties

✓ 1. Collision resistance — given a hash function
it is hard to find **two colliding inputs**

2. Second-preimage resistance — given a hash function and
given a **first** input,
it is hard to find a **second** input
that **collides** with the first

3. Preimage resistance — given a hash function and
given an hash output
it is hard to **invert** that output

Merkle-Damgard construction



<u>MD Theorem</u>: if f is CR, then so is H

M_1	M_2	…	M_m

512 bits

for i **= 1 to** m **do**

$$W_t = \begin{cases} t\text{-th word of } M_i & 0 \le t \le 15 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \ll 1 & 16 \le t \le 79 \end{cases}$$

$A \leftarrow H_0^{i-1};\quad B \leftarrow H_1^{i-1};\quad C \leftarrow H_2^{i-1};\quad D \leftarrow H_3^{i-1};\quad E \leftarrow H_4^{i-1}$

for t **= 1 to 80 do**

$$T \leftarrow A \ll 5 + g_t(B, C, D) + E + K_t + W_t$$

$$E \leftarrow D;\quad D \leftarrow C;\quad C \leftarrow B \gg 2;\quad B \leftarrow A;\quad A \leftarrow T$$

end

160 bits

$H_{0..4}^{i-1}$

$H_0^i \leftarrow A + H_0^{i-1};\quad H_1^i \leftarrow B + H_1^{i-1};\quad H_2^i \leftarrow C + H_2^{i-1};$

$H_3^i \leftarrow D + H_3^{i-1};\quad H_4^i \leftarrow E + H_4^{i-1}$

end

return $H_0^m\ H_1^m\ H_2^m\ H_3^m\ H_4^m$ ← 160 bits

160 bits

Hash Function Security

- Consider best-case scenario (random outputs)
- If a hash function output only 1 bit, how long would we expect to avoid collisions?
  - Expectation: $1\times 0 + 2 \times ½ + 3 \times ½ = 2.5$
- What about 2 bits?
  - Expectation: $1 \times 0 + 2 \times ¼ + 3 \times ¾ ½ + 4 \times ¾ ½ ¾ + 5 \times ¾ ½ ¼ \approx 3.22$
- This is too hard…

Birthday Paradox

- Need another method
  - Birthday paradox: if we have 23 people in a room, the probability is > 50% that two will share the same birthday
    - Assumes uniformity of birthdays
      - Untrue, but this only *increases* chance of birthday match
    - Ignores leap years (probably doesn't matter much)
  - Try an experiment with the class…

Birthday Paradox (cont)

- Let's do the math
  - Let n equal number of people in the class
  - Start with n = 1 and count upward
    - Let NBM be the event that there are **N**o-**B**irthday-**M**atches
    - For n=1, Pr[NBM] = 1
    - For n=2, Pr[NBM] = $1 \times 364/365 \approx .997$
    - For n=3, Pr[NBM] = $1 \times 364/365 \times 363/365 \approx .991$
    - …
    - For n=22, Pr[NBM] = $1 \times \ldots \times 344/365 \approx .524$
    - For n=23, Pr[NBM] = $1 \times \ldots \times 343/365 \approx .493$
  - Since the probability of a match is 1 – Pr[NBM] we see that n=23 is the smallest number where the probability exceeds 50%

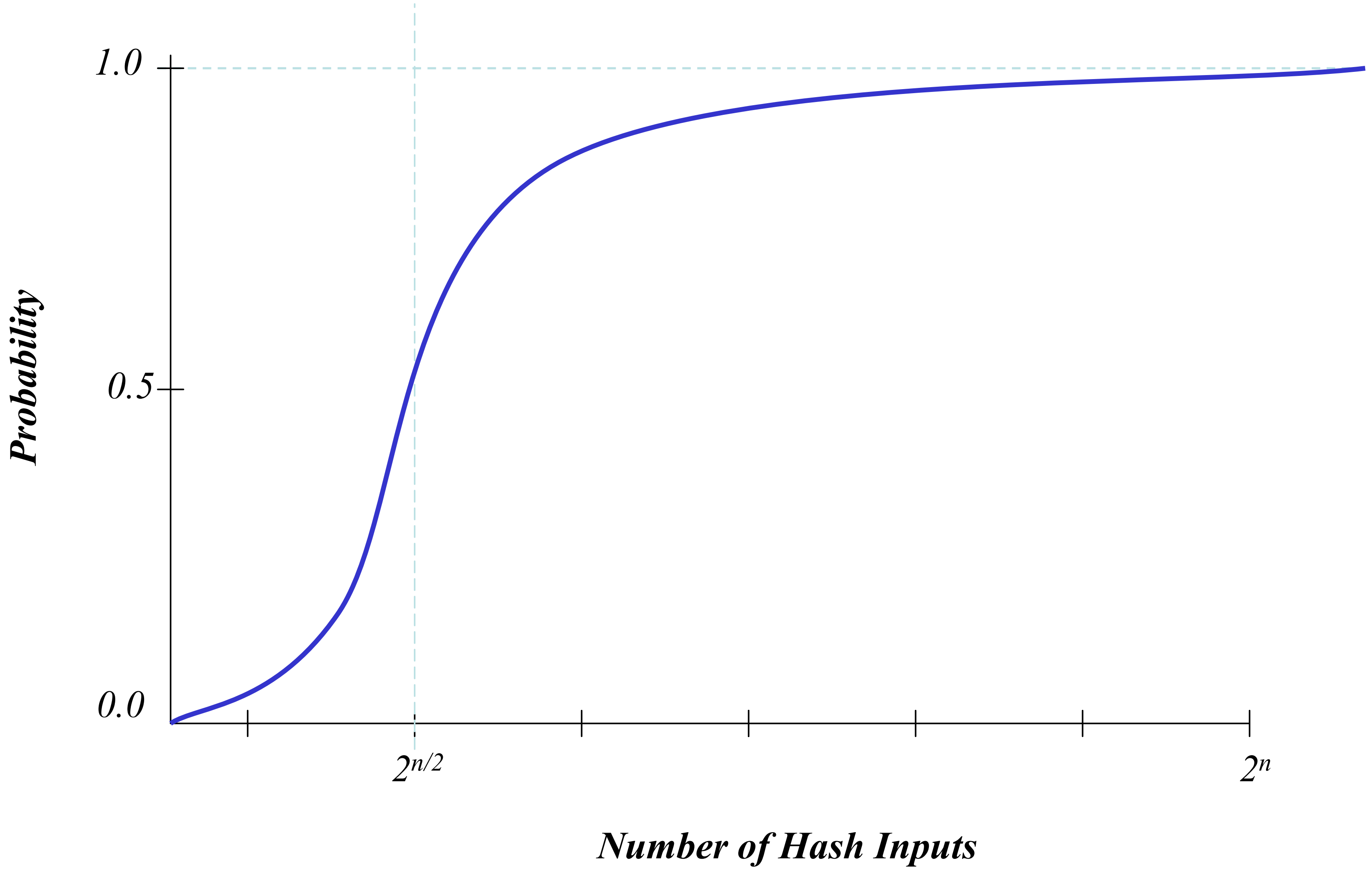
Occupancy Problems

- What does this have to do with hashing?
  - Suppose each hash output is uniform and random on $\{0,1\}^n$
  - Then it's as if we're throwing a ball into one of 2^n bins at random and asking when a bin contains at least 2 balls
    - This is a well-studied area in probability theory called "occupancy problems"
  - It's well-known that the probability of a collision occurs around the square-root of the number of bins
    - If we have 2^n bins, the square-root is $2^{n/2}$

Birthday Bounds

- This means that even a perfect n-bit hash function will start to exhibit collisions when the number of inputs nears $2^{n/2}$
  - This is known as the “birthday bound”
  - It’s impossible to do better, but quite easy to do worse
- It is therefore hoped that it takes $\Omega(2^{64})$ work to find collisions in MD5 and $\Omega(2^{80})$ work to find collisions in SHA-1

The Birthday Bound



Latest News

- At CRYPTO 2004 (August)
  - Collisions found in HAVAL, RIPEMD, MD4, MD5, and SHA-0 (2^{40} operations)
    - Wang, Feng, Lai, Yu
    - Only Lai is well-known
  - HAVAL was known to be bad
  - Dobbertin found collisions in MD4 years ago
  - MD5 news is big!
  - SHA-0 isn’t used anymore (but see next slide)

Collisions in SHA-0

M_1, M_1'

not in SHA-0

$$W_t = \begin{cases} t\text{-th word of } M_i & 0 \le t \le 15 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \ll 1 & 16 \le t \le 79 \end{cases}$$

$A \leftarrow H_0^{i-1}$; $B \leftarrow H_1^{i-1}$; $C \leftarrow H_2^{i-1}$; $D \leftarrow H_3^{i-1}$; $E \leftarrow H_4^{i-1}$

for t **= 1 to ~~80~~ 65 do**

$T \leftarrow A \ll 5 + g_t(B, C, D) + E + K_t + W_t$

$E \leftarrow D$; $D \leftarrow C$; $C \leftarrow B \gg 2$; $B \leftarrow A$; $A \leftarrow T$

end

$H_{0..4}^{i-1}$

$H_0^i \leftarrow A + H_0^{i-1}$; $H_1^i \leftarrow A + H_1^{i-1}$; $H_2^i \leftarrow C + H_2^{i-1}$;

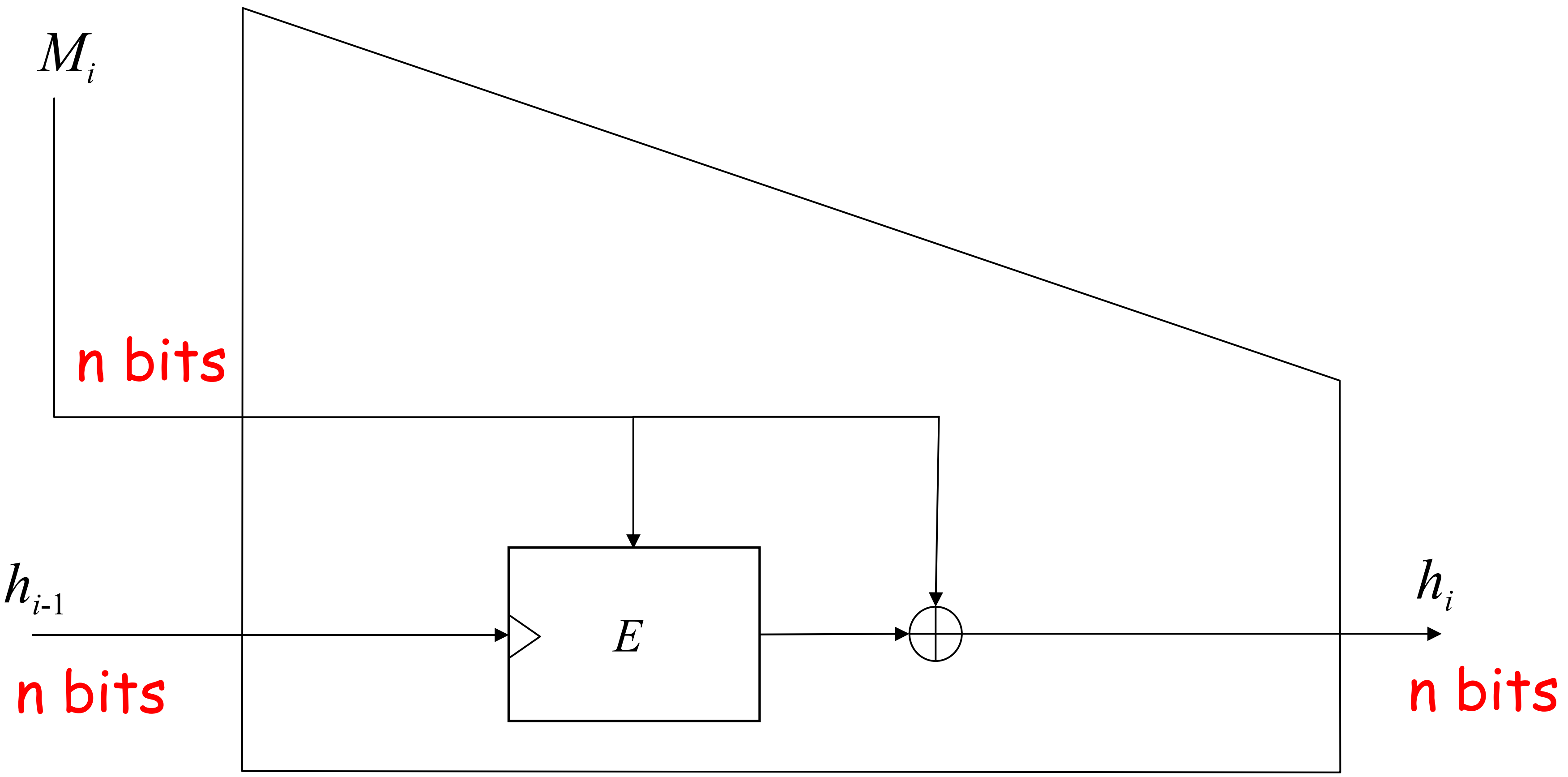
$H_3^i \leftarrow D + H_3^{i-1}$; $H_4^i \leftarrow E + H_4^{i-1}$

Collision!

What Does this Mean?

- Who knows
  - Methods are not yet understood
  - Will undoubtedly be extended to more attacks
  - Maybe nothing much more will happen
  - But maybe everything will come tumbling down?!
- But we have OTHER ways to build hash functions

A Provably-Secure Blockcipher-Based Compression Function

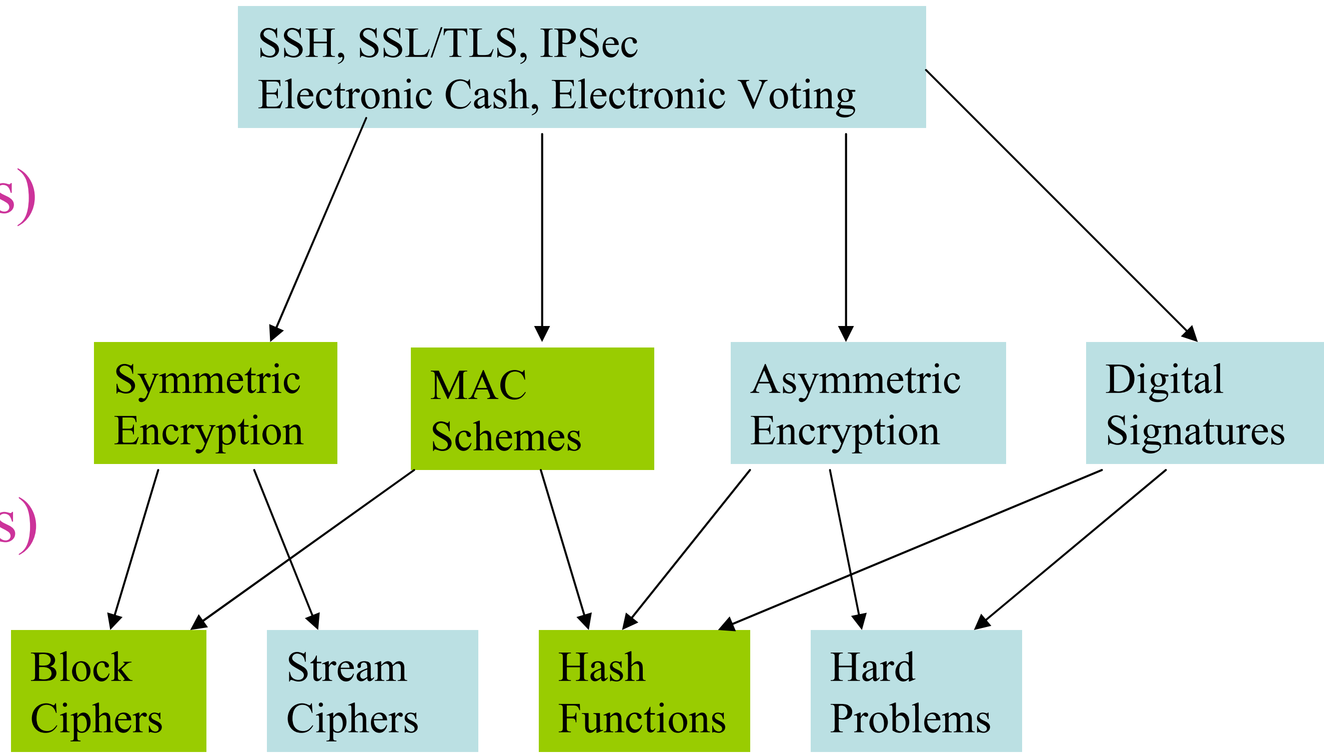


The Big (Partial) Picture

Second-Level Protocols
(Can do proofs)

First-Level Protocols
(Can do proofs)

Primitives



(No one knows how to prove security; make assumptions)