Foundations of Network and Computer Security

John Black

Lecture #5
Sep 7th 2004

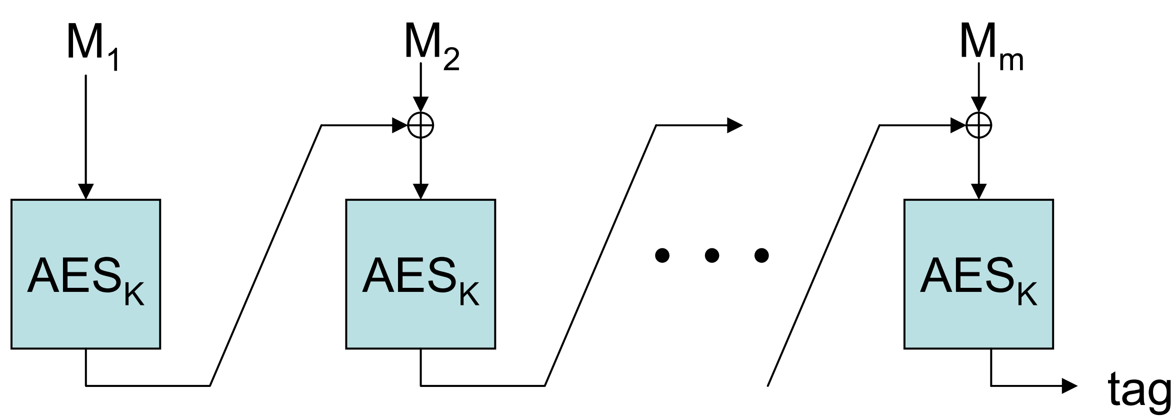
CSCI 6268/TLEN 5831, Fall 2004

Announcements

- Please sign up for class mailing list by end of today
- Quiz #1 will be on Thursday, Day after tomorrow

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



Breaking CBC MAC

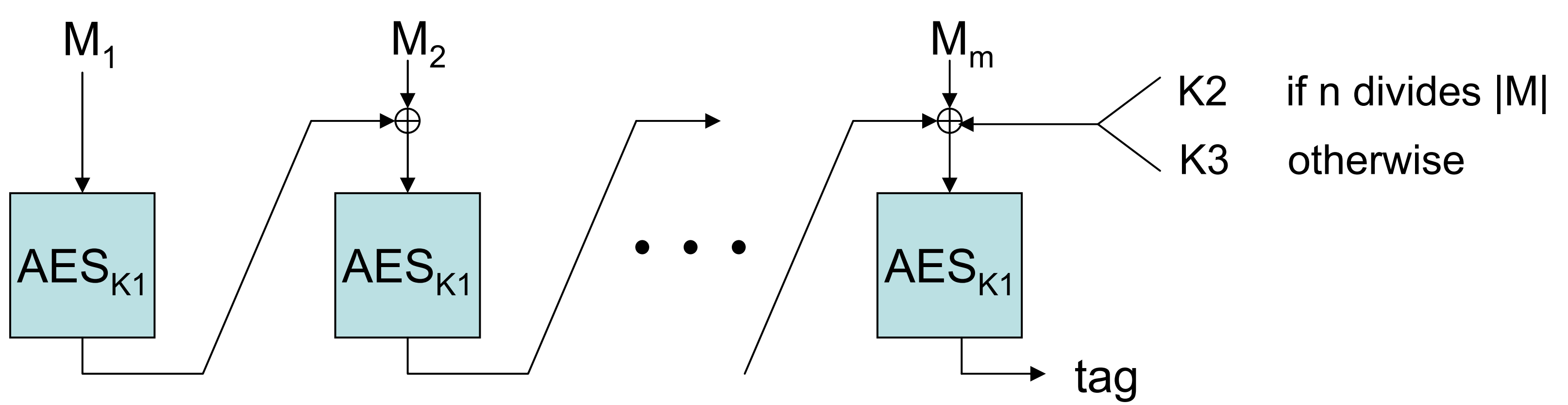
- If we allow msg lengths to vary, the MAC breaks
  - To "forge" we need to do some (reasonable) number of queries, then submit a new message and a valid tag
    - Ask $M_1 = 0^n$ we get $t = AES_K(0^n)$ back
    - We're done!
      - We announce that $M^* = 0^n \,||\, t$ has tag t as well
      - (Note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

CBC MAC Attack (notes)

- Attack was "unfair"
  - We used varying lengths which is not allowed for CBC MAC
  - Well, we were just demonstrating that the fixed-length condition is necessary!
  - And we were giving an example of the ACMA model
- Attack was adaptive
  - We used the output, t, from our first query as a message block in our forgery!
- Forged message wasn't really meaningful
  - Doesn't matter, a forgery is a forgery

Varying Message Lengths: XCBC

- There are several well-known ways to overcome the fixed-length 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")

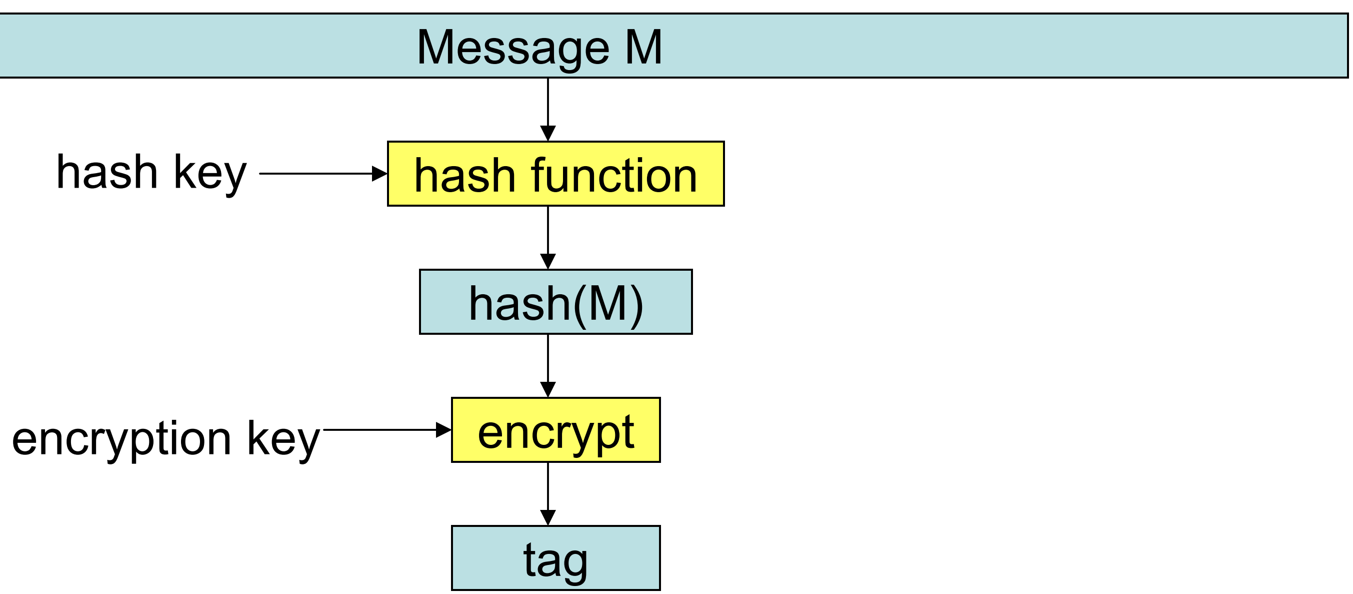


UMAC: MACing Faster

- 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

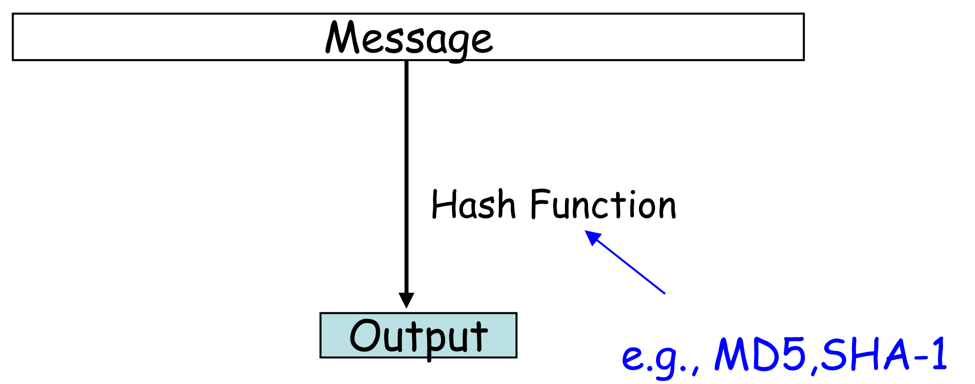
Notes on HMAC

- 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_2	…	M_m

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

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

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

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

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

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

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

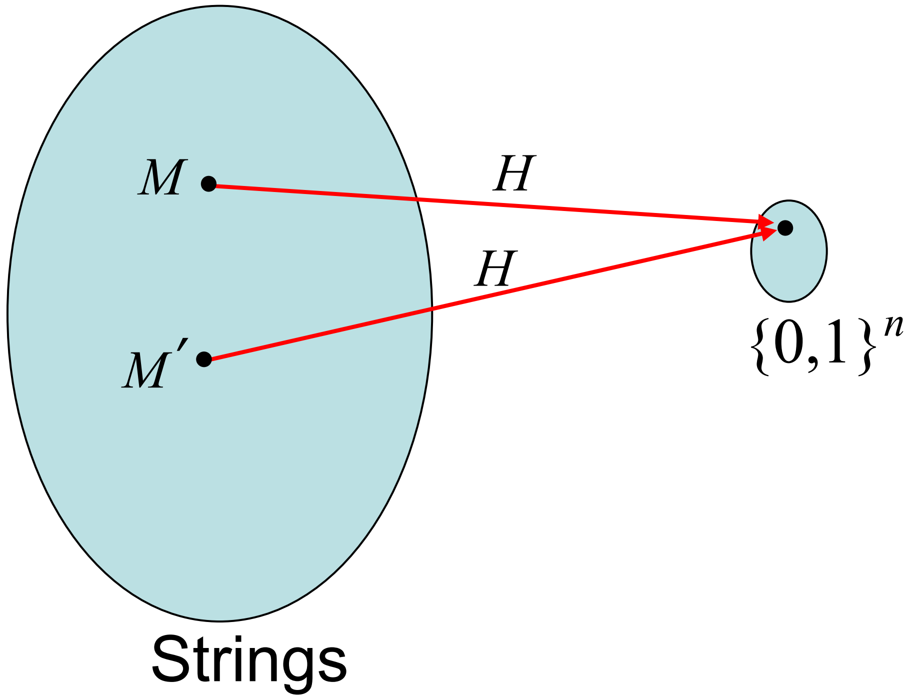
- Message authentication codes (HMAC)
- Digital signatures (hash-and-sign)
- File comparison (compare-by-hash, eg, RSYNC)
- Micropayment schemes
- Commitment protocols
- Timestamping
- Key exchange
- ...

A cryptographic property

(quite informal)

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

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



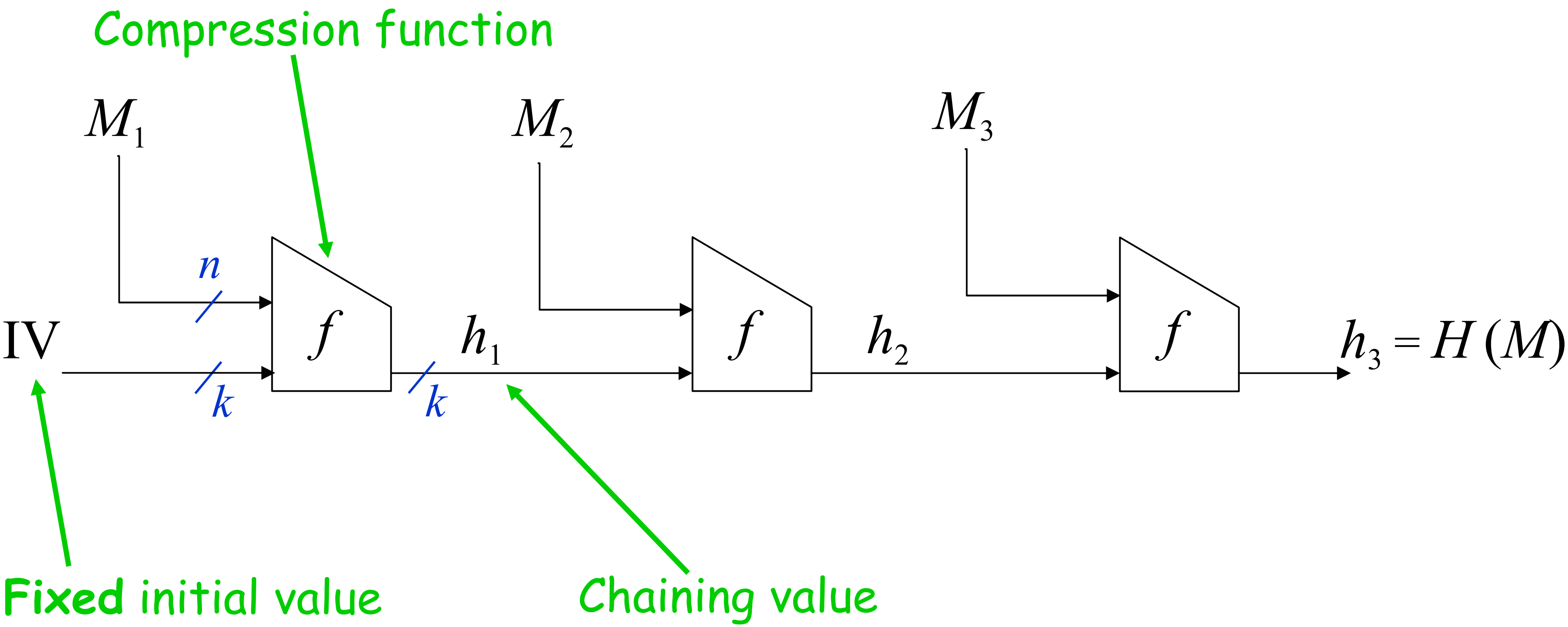
More cryptographic properties

✓ 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**

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

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

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

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

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

end

160 bits

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

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

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: $1 \times 0 + 2 \times \frac{1}{2} + 3 \times \frac{1}{2} = 2.5$
- What about 2 bits?
  - Expectation: $1 \times 0 + 2 \times \frac{1}{4} + 3 \times \frac{3}{4} \frac{1}{2} + 4 \times \frac{3}{4} \frac{1}{2} \frac{3}{4} + 5 \times \frac{3}{4} \frac{1}{2} \frac{1}{4} \approx 3.22$
- 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%

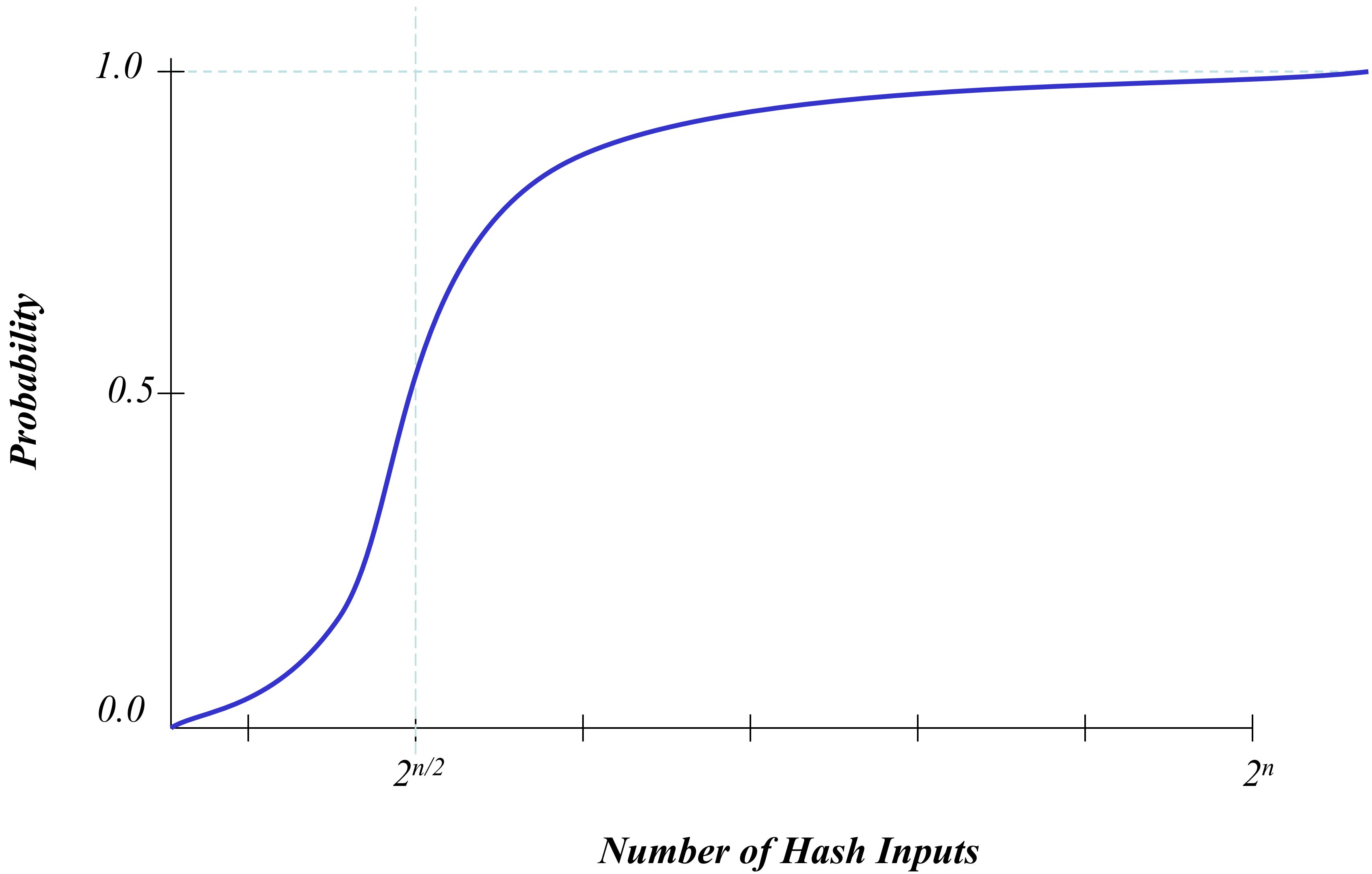
Occupancy Problems

- 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

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

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

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

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

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

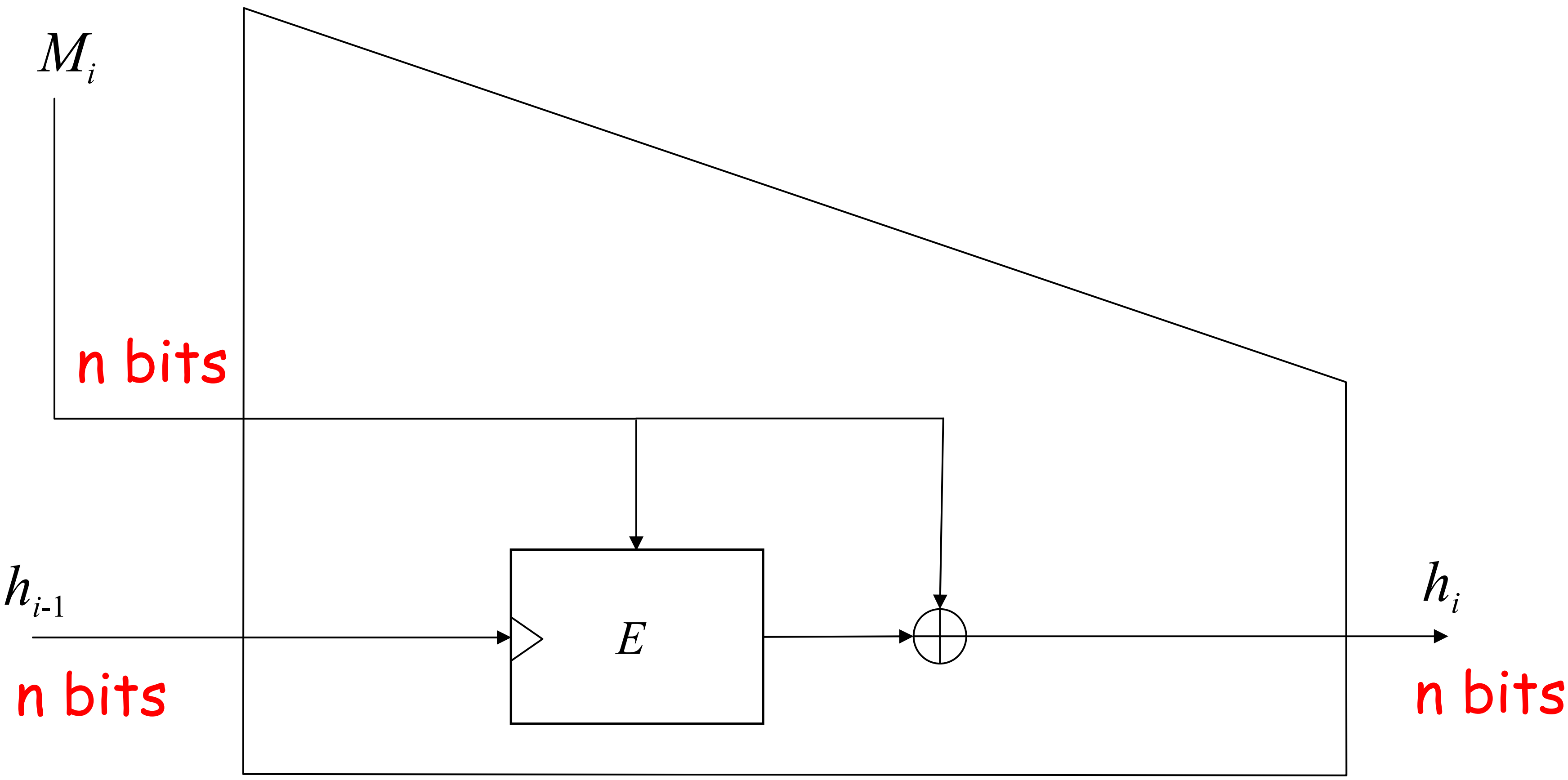
$H_3^i \leftarrow D + H_3^{i-1};\quad H_4^i \leftarrow E + H_4^{i-1}$

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