Lecture 5

Cryptographic Hash Functions

Read: Chapter 5 in KPS

1

1

Purpose

- CHF – one of the most important tools in modern cryptography and security
- CHF-s are used for many authentication, integrity, digital signatures and non-repudiation purposes
- **NOT the same as "hashing"** used in DB or CRCs in communications

2

2

Cryptographic HASH Functions (CHF-s)

- **Purpose:** produce a fixed-size "fingerprint" or **digest** of arbitrarily long input data
- Why? To guarantee **integrity** of input
- Properties of a "good" CHF H():

1. Arbitrary-Length Input
2. Fixed Output
3. Efficiency
4. One-Way-ness
5. Weak Collision-Resistance
6. Strong Collision-Resistance

3

3

CHF-s

Properties of a "good" CHF H():

1. **Arbitrary-Length Input**:
   - Takes on input of any size
2. **Fixed Output**:
   - Produces fixed-length output
3. **Efficiency**:
   - Easy to compute (efficient)

Non-Cryptographic **properties**

4. **One-Way-ness**:
   - Given any h, computationally infeasible to find any x such that H(x) = h
5. **Weak Collision-Resistance**:
   - For a given x, computationally infeasible to find y: H(y) = H(x) and y≠x
6. **Strong Collision-Resistance**:
   - Computationally infeasible to find any (x, y) such that H(x) = H(y) and x ≠ y

Cryptographic **properties**

4

4

Simple Hash Functions

Example: **Bitwise-XOR**

	bit 1	bit 2	. . .	bit *n*
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	.	.	.	.
	.	.	.	.
	.	.	.	.
block *m*	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

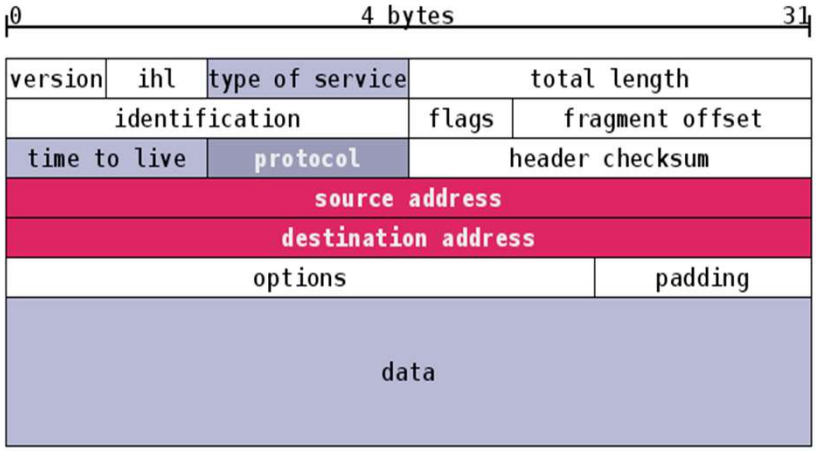
- How to violate desired crypto properties?
- For x, easy to find y that share the same hash code, or H(x) = H(y) --- just need to switch the order of 2 blocks!
- Also, easy to arbitrarily modify the message and append one block that would set the hash code as needed
  - We can create H(m||b) = H(x) = h, where m is arbitrarily chosen, and b = H(m) XOR h

5

5

Another Example

- IPv4 header checksum
- One's complement of the one's complement sum of the IP header's 16-bit words

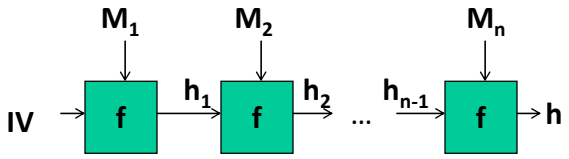


6

6

Construction

- A CHF is typically based on an internal compression function f() that works on fixed-size input blocks (M_i)
  - So-called "Merkle-Damgard" construction:
  - A fixed-size "compression function".
  - Each iteration mixes an input block with the previous block's output



- Sort of like a **Chained Block Cipher**
  - Produces a hash value for each fixed-size block based on (1) its content and (2) hash value for the previous block
  - "Avalanche" effect: 1-bit change in input produces "catastrophic" and unpredictable changes in output

7

7

The Birthday Paradox





- Example hash function: **y=H(x)** where: x=person and H() is Bday()
- y ranges over set Y=[1…365], let n = size of Y, i.e., number of distinct values in the range of H()
- How many people do we need to 'hash' to have a collision?
- Or: what is the probability of selecting **k** random numbers, say x_1, x_2, …, x_k without having $H(x_i) = H(x_j)$, where i <> j ?
- Assuming the mapping from x → H(x) is uniformly distributed, the probability of no collisions is:
  - $P0=1*(1-1/n)*(1-2/n)*\ldots*(1-(k-1)/n)) <= e^{(k(1-k)/2n)}$
    (use $1-x <= e^{-x}$)
- probability of at least one:
  - P1=1-P0
- Set P1 to be at least 0.5 and solve for k:
  - k >= 1.17 * SQRT(n)
  - k >= 22.3 for n=365

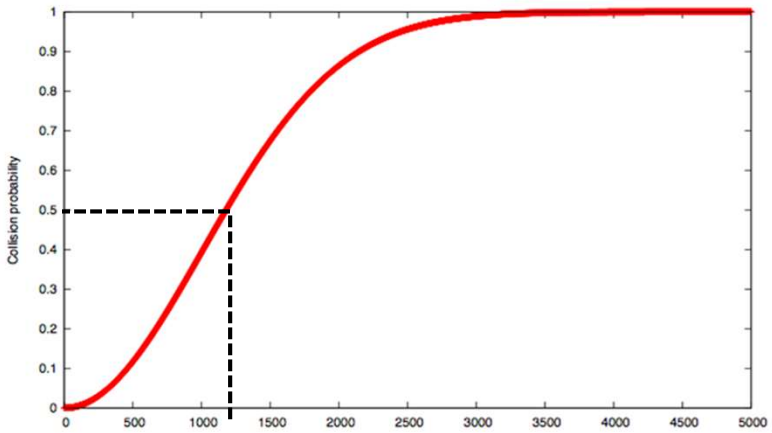
Surprisingly small!

8

8

"Birthday Paradox"

Example: N = 10^6



k = ~1200 to have >= 50% to find a hash collision

9

9

The Birthday Paradox

*Say **n** is the **digest size** of a hash function H(), which is a **m-bit** value ($n=2^m$)*

$\sqrt{2^m} = 2^{m/2}$ trials must be computationally infeasible! Otherwise, finding collisions is easy.

10

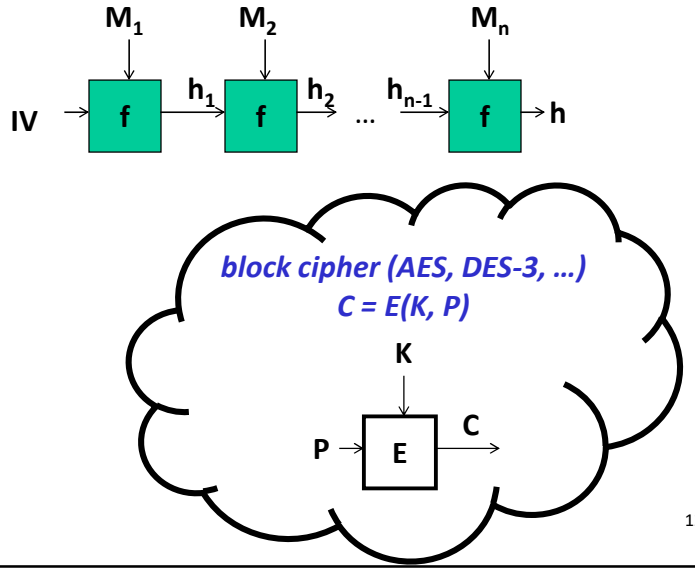
How Long Should a Hash be?

- Many input messages yield the same hash
  - e.g., 1024-bit message, 128-bit hash
  - On average, 2^{896} messages map into 1 hash
- With m-bit hash, it takes about $2^{m/2}$ trials to find a collision (with ≥ 0.5 probability)
- When m=64, it takes 2^{32} trials to find a collision (doable in very little time)
- Today, need **at least** m=160, requiring about 2^{80} trials (180 is better)

11

11

How to construct CHF following the Merkle-Damgard construction?



12

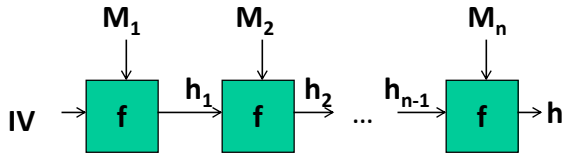
12

CHF from a Block Cipher

One direct option:

- Use block cipher C = E(K, P) as compression func

$$H_i = E(M_i, H_{i-1}), \quad M_o = 0$$

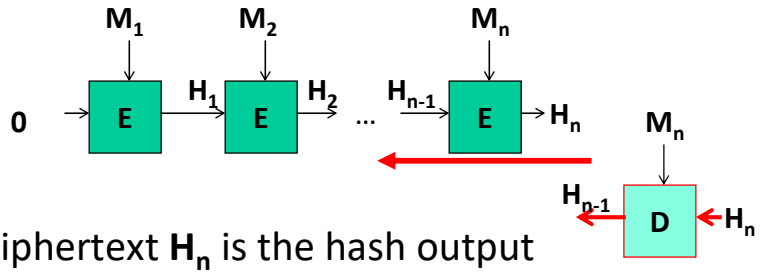


CHF from a Block Cipher

One direct option:

- Use block cipher C = E(K, P) as compression func

$$H_i = E(M_i, H_{i-1}), \quad M_o = 0$$



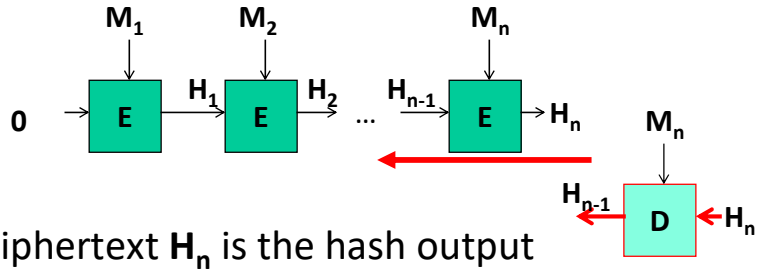
- Final ciphertext H_n is the hash output
- Secure?

CHF from a Block Cipher

One direct option:

- Use block cipher C = E(K, P) as compression func

$$H_i = E(M_i, H_{i-1}), \quad M_o = 0$$



- Final ciphertext H_n is the hash output
- Secure?

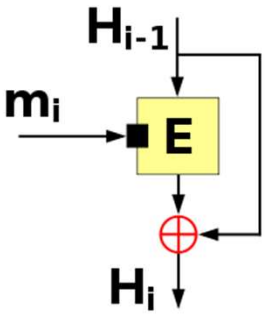
One-Way-ness: Given any h, computationally infeasible to find any x such that H(x) = h

15

CHF from a Block Cipher

Davies-Meyer CHF:

- $H_i = H_{i-1} \oplus E(M_i, H_{i-1})$, H_o=0
- Compression function is secure if is a secure block cipher



16

16

CHF Examples

	MD5 (defunct!)	**SHA-1 (weak!)**	**SHA-256 (SHA-2 family, used today)**
Digest length	128 bits	160 bits	256 bits
Block size	512 bits	512 bits	512 bits
# of steps	64	80	64
Max msg size	∞	2^{64}-1 bits	2^{64}-1 bits
Security against collision attacks	<=18 bits	<= 63 bits	128 bits

17

17

Latest standard: SHA-3

- Public competition by NIST, similar to AES:
- NIST request for proposals (2007)
- 51 submissions (2008)
- 14 semi-finalists (2009)
- 5 finalists (2010)
- Winner: Keccak (2012)
  - Designed by Bertoni, Daemen, Peeters, Van Assche.
  - Based on “sponge construction”, a completely different structure from prior CHF-s.

18

18

What are hash functions good for (besides integrity)?

19

19

Message Authentication Using a Hash Function

Use symmetric encryption (AES or 3-DES) and a hash function

- Given message M
- Compute H(M)
- Encrypt H(M) in ECB or CBC mode
- Result is: $E_K(H(M))$ = MAC
- Alice sends to Bob: MAC, M
- Bob receives MAC',M' decrypts MAC' with K, hashes result and checks if: $D_K(MAC')$ =?= H(M')

Collision ➔ MAC forgery!

20

20

Using Hash for Authentication

Alice and Bob share a secret key K_{AB}

1.Alice ➔ Bob: random challenge r_A

2.Bob ➔ Alice: $H(K||r_A)$, random challenge r_B

3.Alice ➔ Bob: $H(K||r_B)$

Only need to compare H() results

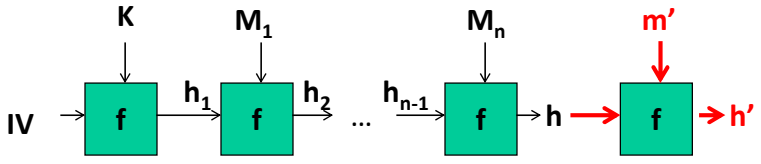
21

21

Using Hash to Compute a MAC: message **integrity+authenticity**

- Just computing and appending H(m) to m is enough for **integrity** but **not for authenticity**
- Need a "Keyed Hash":
  - Prefix:
    - MAC: H(K || m), almost works, but …
    - Allows concatenation with arbitrary message:

H(K || m || m')



22

22

Using Hash to Compute a MAC: message **integrity+authenticity**

- Just computing and appending H(m) to m is enough for **integrity** but **not for authenticity**
- Need a "Keyed Hash":
  - Prefix:
    - MAC: H(K || m), almost works, but …
    - Allows concatenation with arbitrary message:

      H(K || m || m')
  - Suffix:
    - MAC: H(m || K)
    - Works better, but what if m' is found such that H(m)=H(m')?
  - HMAC:
    - H (K || H (K || m))

23

23

Hash Function-based Keyed MAC (HMAC)

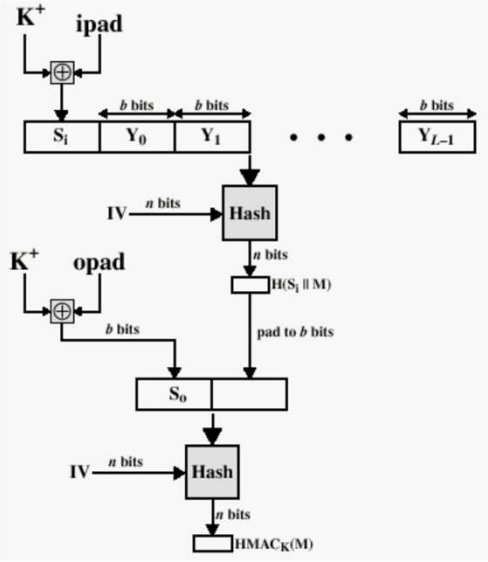
- **Main Idea**: Create a MAC derived from any CHF
  - CHF do not use a key, therefore cannot be used directly as a MAC
- **Motivations for HMAC:**
  - Cryptographic hash functions run faster in software than many encryption algorithms such as 3-DES
  - No need for the function to be reversible
  - No US Government export restrictions (was important in the past)
- **Status**: designated as mandatory for IP security
  - Also used in TLS, IPSec, etc.

24

24

HMAC Algorithm

- Compute H1 = H() of the concatenation of M and K1
- To prevent an "additional block" attack, compute again H2= H() of the concatenation of H1 and K2
- Notation:
  - K^+ = K padded with 0's
  - ipad=00110110 x b/8
  - opad=01011100 x b/8
- Execution:
  - Same as H(M), plus 2 blocks



Essentially H (K || H (K || m))

25

25

CHF Examples

	MD5 (defunct!)	**SHA-1 (weak!)**	**SHA-256 (SHA-2 family, used today)**
Digest length	128 bits	160 bits	256 bits
Block size	512 bits	512 bits	512 bits
# of steps	64	80	64
Max msg size	∞	2^{64}-1 bits	2^{64}-1 bits
Security against collision attacks	<=18 bits	<= 63 bits	128 bits

26

26