CSCI361

Computer Security

Hashing

Outline

- Hash functions.
  - Cryptographic requirements.
- Signing with hash functions.
- Assessing security.
  - The birthday attack.
- Techniques for hash functions.
  - Block ciphers.
  - Modular arithmetic.
  - Dedicated or designed.

Hash functions

- A *hash function* **H** accepts a message **X** of arbitrary size and outputs a block of fixed size, called a *message digest* or *hash value*. The idea is that, in general, **H(X)** is much smaller than **X**.
  - **Example**: The message is an arbitrary file and the digest is a 128 bit block.

- It is possible that two distinct messages produce the same message digest. This is called a *collision*.
  - **Example**:

    H(X) = X mod 10

    H(56) = H(96) = H(156)
- For cryptograph we require hash functions with *collision resistance.* That is, where it is computationally difficult to find two inputs **x** and **y** such that **H(x)=H(y)**.

Cryptographic hash functions

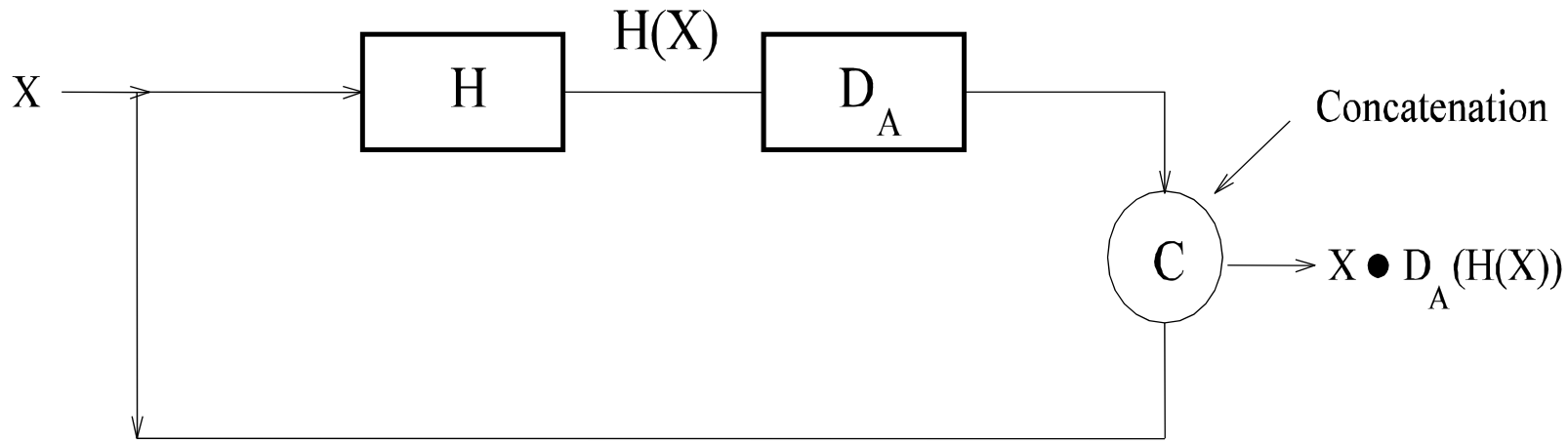
- We require the following properties for cryptographic hash functions:

1. It can be applied to any size input.
2. The output must be of fixed size.
3. **One-way**: Easy to calculate but hard to invert.
4. **Pre-image resistant**: For any given **Y**, it is difficult to find an **X** such that **H(X)=Y**.
5. **Second Pre-image resistant**: Given $\mathbf{X_1}$ it should be difficult to find another $\mathbf{X_2}$ such that $\mathbf{H(X_1)=H(X_2)}$.
6. **Collision resistant**: It is computationally infeasible to find messages **X** and **Y** with **X≠Y** such that **H(X)=H(Y)**.

Properties 3 and 4 are closely related.

Signing with hash functions

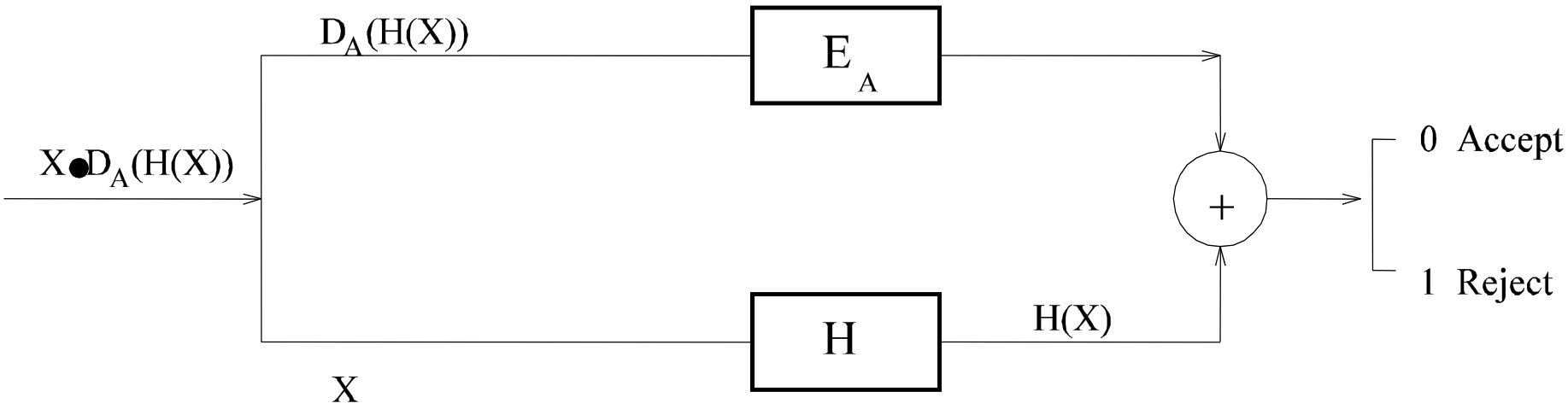
- Hash functions can be used as a means to provide message integrity. One of the main applications of hash function is for digital signatures.
- To sign a message **X**, the hash value **H(X)** is calculated and signed.



Signature Generation

Verification

- Verification involves calculating the hash on the message again and comparing it with the transmitted hash.



Signature Verification

- In the context of signatures;
  - Cryptographic hash properties 1, 2 and 3 are required for efficient signature generation.
  - Cryptographic hash properties 4,5 and 6 are required to stop attackers forging signatures.

- Consider that **O**scar, a malicious person, can generate **X≠X'** with **H(X)=H(X')** such that Alice is happy to sign **X** and produce $\mathbf{S_A(X)}$. Now **O**scar can use $\mathbf{(X', S_A(X))}$ as a signed message.

- For property 4, 5 and 6 to hold, the size of the output space must be large, currently the standard is 2^{160}. That is, the hash value must be at least 160 bits.

- A strong signature scheme and a secure hash function may produce a weak signature scheme. An example of this was demonstrated This problem is shown by Coppersmith in combining RSA and the CCITT X.509 hash function.

Assessing the security of hash functions

- If the size of message space is bigger than digest space, we can always find collisions (the pigeonhole principle).
- **Example**:
  - Let messages belong to $\mathbf{Z_p^*}$**= {1,2, …, p-1}**, and digests belong to $\mathbf{Z_q^*}$**={1,2, …, q-1}** where **q** is a prime and **p>q**.

- Choose a number $\mathbf{g} \in \mathbf{Z_q^*}$.
- We define a hash function $h(x)= g^x \bmod q$.

- Let q=15, p=11, and choose g=3.
- We have,

  $3^2 = 9 \bmod 11$ $\quad\quad$ $3^3 = 5 \bmod 11$

  $3^4 = 4 \bmod 11$ $\quad\quad$ $3^5 = 1 \bmod 11$

  $3^6 = 3 \bmod 11$ $\quad\quad$ $3^7 = 9 \bmod 11$

  ➔Collision between x=2 and x=7.

- If 4 bits are used to represent messages for this system, then,

  h(0010)=h(0111)

- Obviously this is not a good cryptographic hash function. How we do measure the likelihood of collisions, and thus the security?

Birthday Attack

- This terminology arises from the **Birthday paradox**.
- **Question**:
  - What is the smallest size class such that the chance of two students having the same birthday is *at least ½?*

- The answer is surprisingly low: there need to be only 23 students in the class. This is only a paradox in the sense it seems counter-intuitively low.

- This can be applied to finding collisions in hash functions, or at least to calculating the probability of finding collisions.

- Suppose there are **m** possible hash values (message digests). Assume the associated with the same (or similar) number of messages. If we evaluate the hash value of **k** randomly selected messages, the probability of at least one collision is

$$P(m,k) > 1 - e^{\frac{-k(k-1)}{2m}} = \varepsilon$$

- Note that the chance of success in this attack depends only on
  - The size of the message digest, and
  - The number of messages digests are calculated for.
- In particular, it does not depend on the specific hash function used. That is the size of digest space puts a lower bound on the probability of success.

$$k = \sqrt{2m \ln\left(\frac{1}{1-\varepsilon}\right)}$$

- Calculate this for ε=0.5, m=365… k=22.49…
- The formula suggests that with sqrt(n) evaluations of the hash function there is a good chance (about 50%) of a collision being found.
- We can use this lower bound to find the required size of digest space, that is the number of bits of the digest, if we assume certain values for the level of feasible computation.

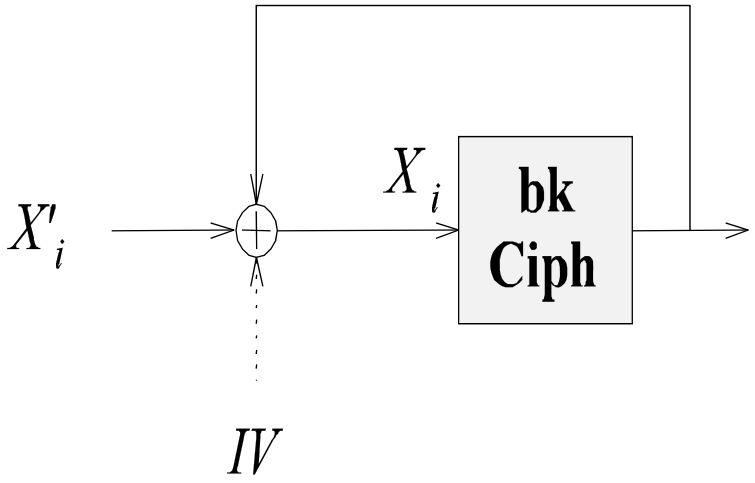
$$m = \frac{k^2}{2\ln\left(\frac{1}{1-\varepsilon}\right)}$$

Techniques for hashing

- Hashing can be divided into three main categories:
  1. Techniques based on block ciphers (symmetric key).
  2. Techniques based on modular arithmetic.
  3. Dedicated or designed hash functions (Others).

Hash functions from block ciphers

- Using block ciphers to produce secure hash functions is not easy. Many schemes have been proposed and broken.
- The simplest way is to use a block cipher in cipher-block chaining mode.



There is only output at the end!

- This is actually a message authentication code (CBC-MAC), where the key must be known to both sides.

$$X = X_1, X_2, ..., X_n$$
$$Y_i = E_K(X_i \oplus Y_{i-1})$$
$$H(X) = Y_n$$

- This method is standardized for banking authentication (ANSI 9.9. ANSI 9.19 ISO 873-1).

- **Disadvantages**:
  - The recipient must have the key;
  - If the key is known it is easy to forge messages, that is, it doesn't actually provide authentication.

- Actually, for any **n**-bit block **Y** and any sequence of **n**-bit blocks $X_1, X_2, \ldots, X_w$ it is possible to choose a further **n**-bit block X_{w+1} such that

  $$X_{w+1} = D_K(Y) \oplus Y_w$$

  where Y_w is the hash value of $X_1, X_2, \ldots, X_w$.
- This new message with w+1 blocks has exactly the same hash value since the xor's cancel out.

Hash functions based on modular arithmetic: An example.

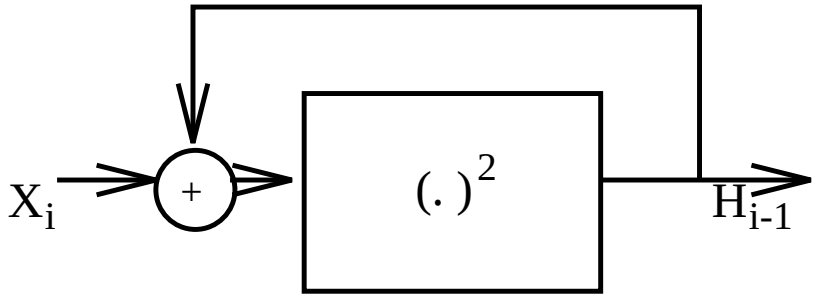
- Quadratic Congruential Manipulation Detection Code (QCMDC) (Jueneman (1983))

- This scheme is a *keyed hash function*.
- Break the message into fixed size blocks of **m**-bits. $\mathbf{H_0}$ is a randomly chosen *secret initial value* (the key).
  - **M** is a prime satisfying $\mathbf{M} \geq \mathbf{2^{m-1}}$.
  - + is integer addition.
- Then

$$H_i = (H_{i-1} + X_i)^2 \mod M$$

- H_n is the hash value.

- This scheme is broken. (Coppersmith)}

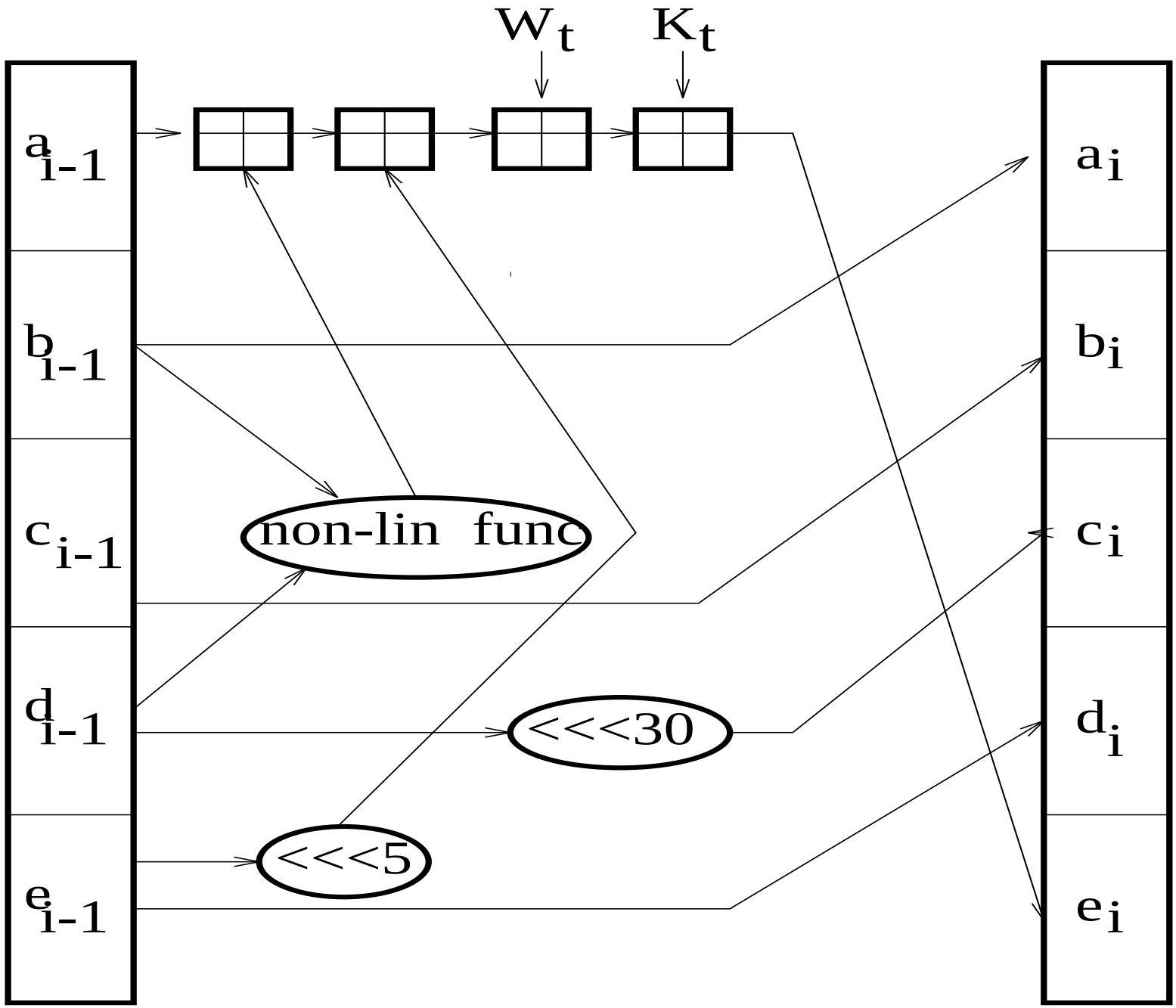


Designed hash functions

- MD5 (Rivest 1992).
- One of the best known hash algorithms.
  - It processes the input as blocks of 512-bit and produces a 128-bit message digest.
  - MD5 is the successor of MD4. It gave up a little in speed for a much greater likelihood of ultimate security.
  - The MD5 algorithm is designed to be fast on 32-bit machines.
  - It uses simple operations such as *addition modulo 32*, and can therefore be coded quite compactly and exexcuted very quickly.
- Collisions were found against MD5 in 2004, that is a method of finding collisions in a practical time.
  - As early as 1993 partial collisions were found.
- Significantly faster methods of finding collisions were demonstrated in 2005.

Secure Hash Algorithm (SHA-1)

- A hash algorithm proposed and adopted by NIST, for use with the DSA standard.
- Kind of a successor to MD5.
- Designed as an improvement to SHA-0.
- It produces a 160 bit message digest and uses the design approach used in MD5.
- On the next page we show a single SHA-1 operation.
- The plus boxes are addition mod 2^{32}.
- The non-linear function and $\mathbf{K_t}$ vary for different operations.


W t
K t
a i-1
b i-1
c i-1
d i-1
e i-1
non-lin func
<<<30
<<<5
a i
b i
c i
d i
e i

- In SHA-1 the message is padded to make 512-bit blocks (this is also done in MD5).
- The algorithm processes 512 bit blocks in each round. Each round has 4 sub-rounds and each sub-round consists of 20 operations. The nonlinear function used in each round is different.
- $\mathbf{W_t}$**, i=0,1 … 79** are 32-bit blocks constructed from a 512 bit message blocks.
- $\mathbf{K_t}$ is a constant dependent upon the sub-round.
- SHA-1 is broken, but not yet to a practical level for attackers except for large scale distributed systems. The older SHA-0 is thoroughly broken and is insecure.
- No attacks on SHA-2 (variants) have been reported so far.

Other hash functions

- HAVAL.
- RIPEMD-(160).
- Snefru.
- Tiger.
- WHIRLPOOL.

MAC Based on a Keyed Hash Function

- Hash functions can be used to construct message authentication codes (MAC).
- In this application the sender and receiver share a key **K**.

- A common way of constructing a MAC from a hash function is:

$$\text{HMAC} = H(K||M||K)$$

- Here HMAC denotes **H**ash-based **MAC**. The key **K** is pre-fixed and post-fixed to the message. There are various variations on this.

Latest Progress

- SHA-3 Competition
- Round 1: NIST announced 14 Second Round Candidates on July 24, 2009, and completed the first round of the SHA-3 Competition.
- http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/index.html

Final Round

- Finalist:
  - ***BLAKE***
  - ***Grøstl***
  - ***JH***
  - ***Keccak***
  - ***Skein***

Winner: SHA-3 Algorithm

- NIST announced ***KECCAK*** as the winner of the SHA-3 Cryptographic Hash Algorithm Competition and the new SHA-3 hash algorithm in a press release issued on October 2, 2012.

Winner: SHA-3 Algorithm

- KECCAK was designed by a team of cryptographers from Belgium and Italy, they are:
  - Guido Bertoni (Italy) of STMicroelectronics,
  - Joan Daemen (Belgium) of STMicroelectronics,
  - Michaël Peeters (Belgium) of NXP Semiconductors, and
  - Gilles Van Assche (Belgium) of STMicroelectronics.
- http://csrc.nist.gov/groups/ST/hash/sha-3/winner_sha-3.html