

UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Introduction to Cryptography
ECE 597XX/697XX

Part 11

Hash Functions

Israel Koren

ECE597/697 Koren Part.11 .1

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Content of this part

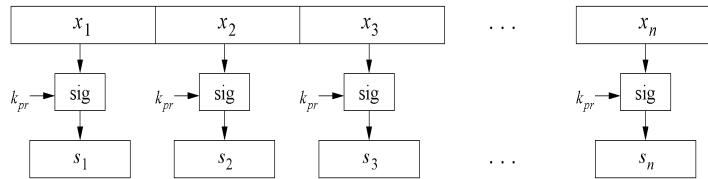
- ◆ Why do we need hash functions?
- ◆ How do they work?
- ◆ Security properties
- ◆ Algorithms
 - Based on block ciphers
 - Other
- ◆ Example: The Secure Hash Algorithm SHA-1
- ◆ SHA-3

ECE597/697 Koren Part.11 .2

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Motivation

Problem: Naive signing of long messages generates a signature of same length.



• 3 Problems

- Computational overhead
- Message overhead
- Security issues

Solution:

Instead of signing the whole message, sign only a digest (=hash).

Also secure, but much faster

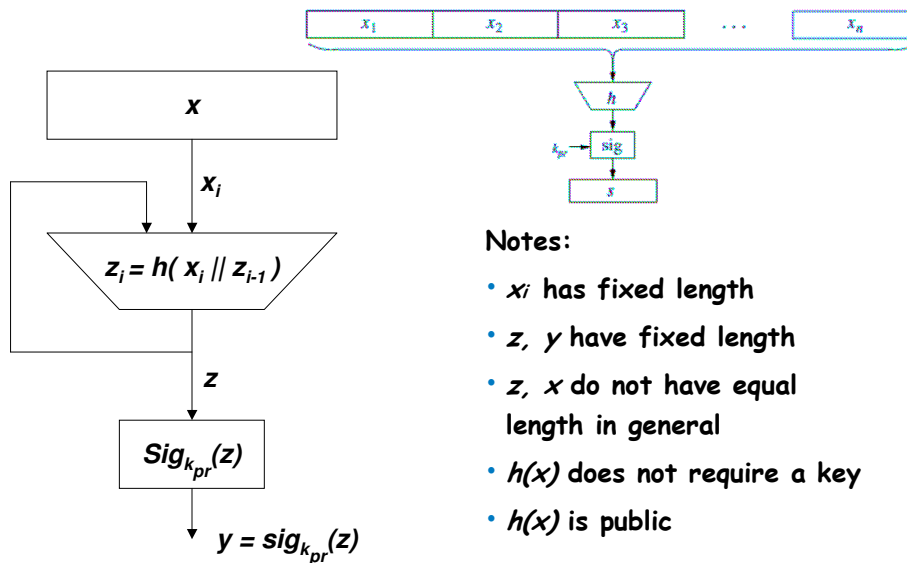
Needed:

Hash Functions

ECE597/697 Koren Part.11 .3

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Digital Signature with a Hash Function



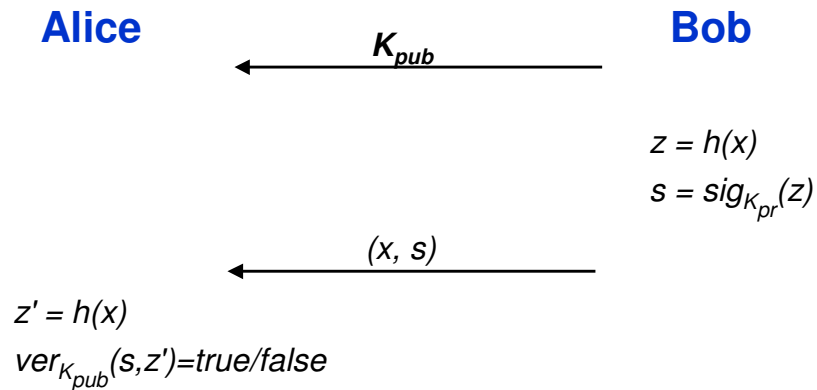
Notes:

- x_i has fixed length
- z , y have fixed length
- z , x do not have equal length in general
- $h(x)$ does not require a key
- $h(x)$ is public

ECE597/697 Koren Part.11 .4

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

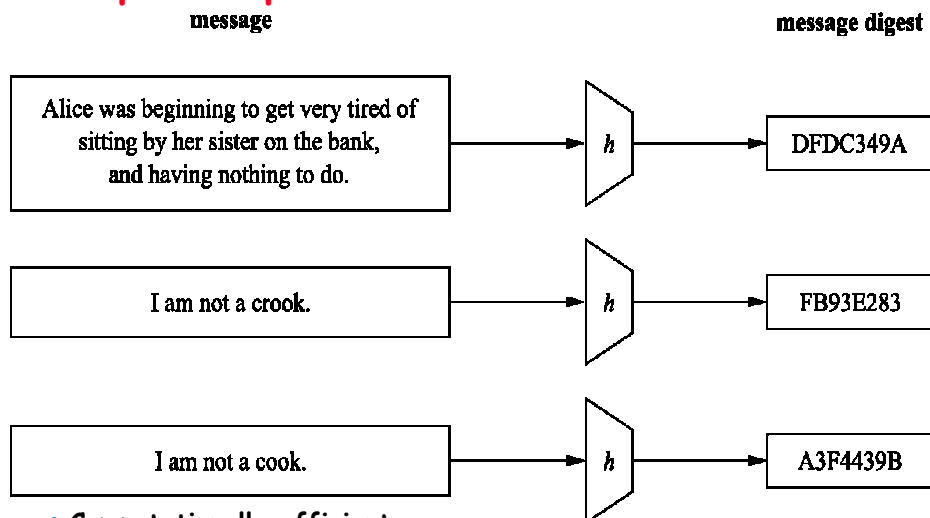
Basic Protocol for Digital Signatures with a Hash Function



ECE597/697 Koren Part.11 .5

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Input-output behavior of hash functions



- Computationally efficient
- Fixed-length output
- Highly sensitive to input

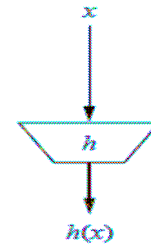
ECE597/697 Koren Part.11 .6

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Security properties of hash functions

1. Preimage resistance - one-wayness

- **Preimage resistance:** For a given output z , it is impossible to find an input x such that $h(x) = z$, i.e., $h(x)$ is one-way.



- **Bob** sends $(e_K(x), \text{sig}_{K_{pr,B}}(z))$

Encrypts with AES and signs with RSA: $s = \text{sig}_{K_{pr,B}}(z) = z^d \bmod n$

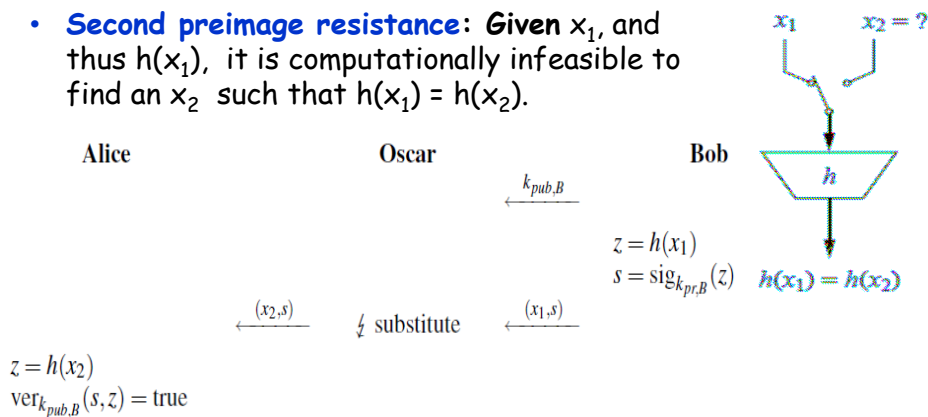
- **Oscar** uses Bob's public key to calculate $s^e = z \bmod n$
If $h(x)$ is not one-way then $x = h^{-1}(z)$

ECE597/697 Koren Part.11 .7

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

2nd security properties of hash functions

- **Second preimage resistance:** Given x_1 , and thus $h(x_1)$, it is computationally infeasible to find an x_2 such that $h(x_1) = h(x_2)$.



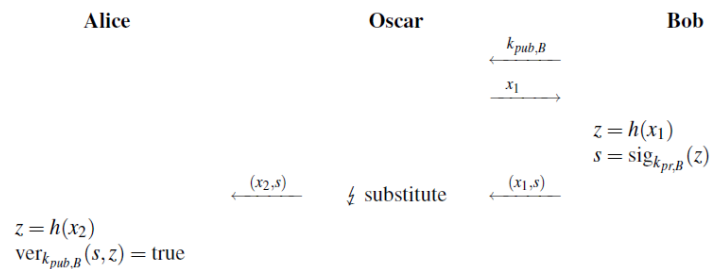
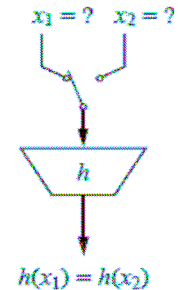
- There is always x_2 such that $h(x_1) = h(x_2)$ but it should be difficult to find
- "weak" collision, requires exhaustive search

ECE597/697 Koren Part.11 .8

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

3rd security properties of hash functions

- **(Strong) Collision resistance:** It is computationally infeasible to find a pair $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$.
- x_1 = "transfer \$10 to Oscar's account"
- x_2 = "transfer \$10,000 to Oscar's account"



ECE597/697 Koren Part.11 .9

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Hash function: Security

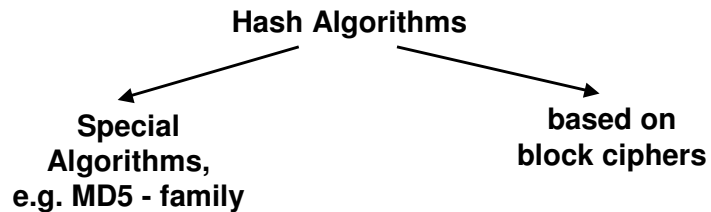
It turns out that collision resistance is more difficult to achieve

- How hard is it to find a collision with a probability of 0.5 ?
- Related Problem: How many people are needed such that two of them have the same birthday with a probability of 0.5 ?
- No! Not $365/2=183$. 23 are enough. This is called the birthday paradox (search takes $\approx \sqrt{2^n}$ steps)
- To deal with this paradox, hash functions need a output size of at least 160 bits

ECE597/697 Koren Part.11 .10

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Hash function: Algorithms

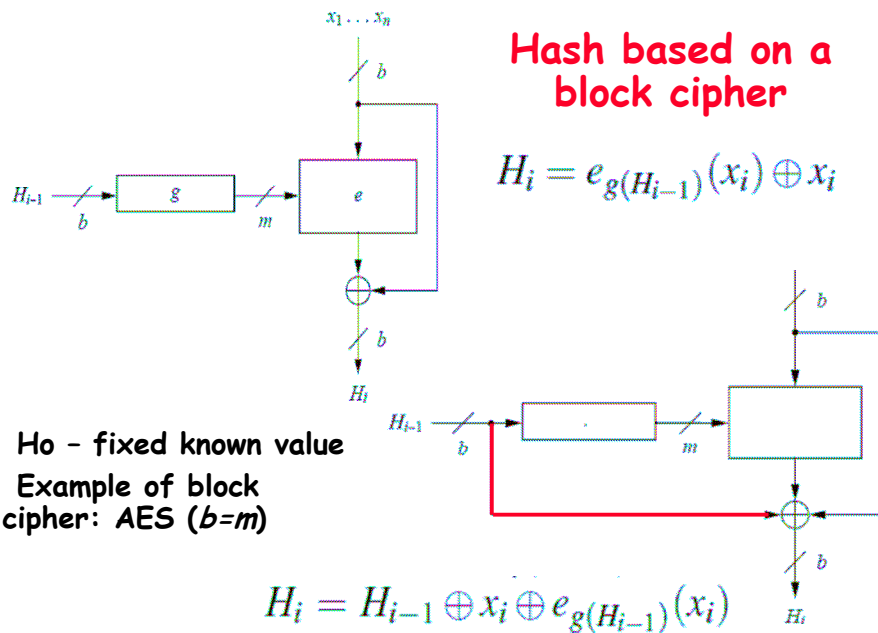


- MD4 and MD5 - families of Hash functions
- SHA-1: output - 160 Bit; input - 512 bit chunks of message x ; operations - bitwise AND, OR, XOR, complement and cyclic shifts.
- RIPE-MD 160: output - 160 Bit; input - 512 bit chunks of message x ; operations - like in SHA-1, but two in parallel and combinations of them after each round.

ECE597/697 Koren Part.11 .11

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

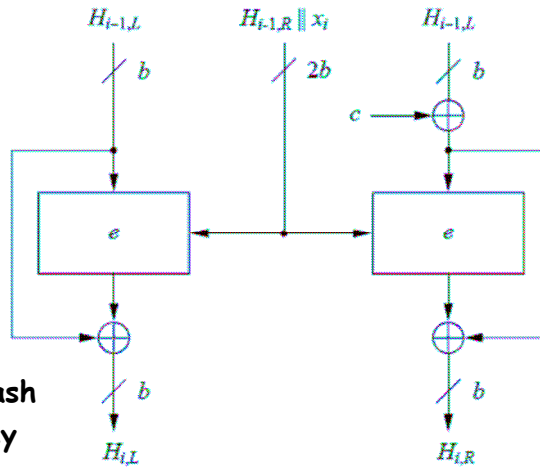
Hash based on a block cipher



ECE597/697 Koren Part.11 .12

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Block cipher based Hash with $m=2b$



- ♦ Twice the block size
- ♦ Example - **Hiroshi** Hash
- ♦ AES with 256-bit key
- ♦ c - non-zero constant

ECE597/697 Koren Part.11 .13

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-1

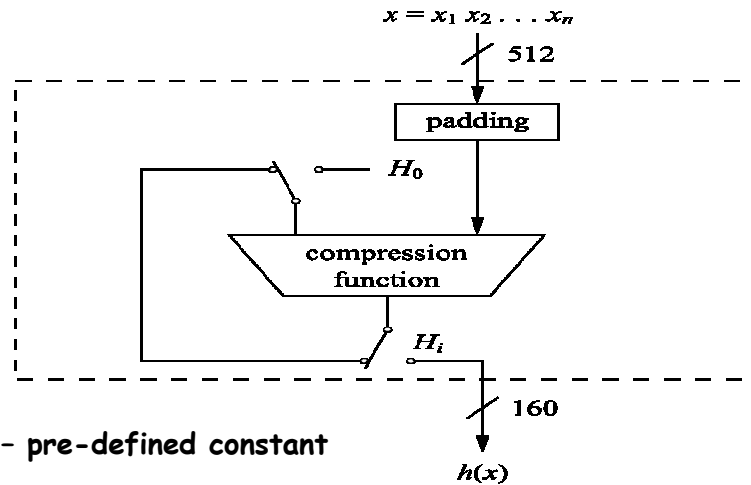
- ♦ Part of the MD-4 family.
- ♦ Based on a Merkle-Damgård construction.
- ♦ 160-bit output from a message of maximum length 2^{64} bit.
- ♦ Widely used (even though some weaknesses are known)

ECE597/697 Koren Part.11 .14

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-1 High Level Diagram

- ◆ Compression function consists of 80 rounds which are divided into four stages of 20 rounds each



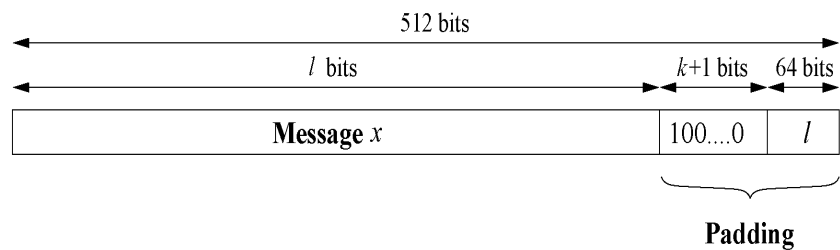
- ◆ H_0 - pre-defined constant

ECE597/697 Koren Part.11 .15

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-1: Padding

- ◆ Message x has to be padded to fit a size of a multiple of 512 bit.
- ◆ $k \equiv 512 - 64 - 1 - l = 448 - (l + 1) \bmod 512$.
- ◆ 64 bits for the binary representation of l



ECE597/697 Koren Part.11 .16

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-1: Hash Computation

- Each message block x_i is processed in four stages with 20 rounds each (512 bits producing 160=32x5 bits)

SHA-1 uses:

- A message schedule which computes 32-bit words $W_0, W_1, \dots, W_{79}$ for each of the 80 rounds
- Five working registers of size of 32 bits A, B, C, D, E
- A hash value H_i consisting of five 32-bit words $H_i^{(0)}, H_i^{(1)}, H_i^{(2)}, H_i^{(3)}, H_i^{(4)}$
- In the beginning, the hash value holds the initial value H_0 , which is replaced by a new hash value after the processing of each single message block.
- The final hash value H_n is equal to the output $h(x)$ of SHA-1.

$$A = H_0^{(0)} = 67452301$$

$$B = H_0^{(1)} = \text{EFCDA889}$$

$$C = H_0^{(2)} = 98BADCFE$$

$$D = H_0^{(3)} = 10325476$$

$$E = H_0^{(4)} = \text{C3D2E1F0}$$

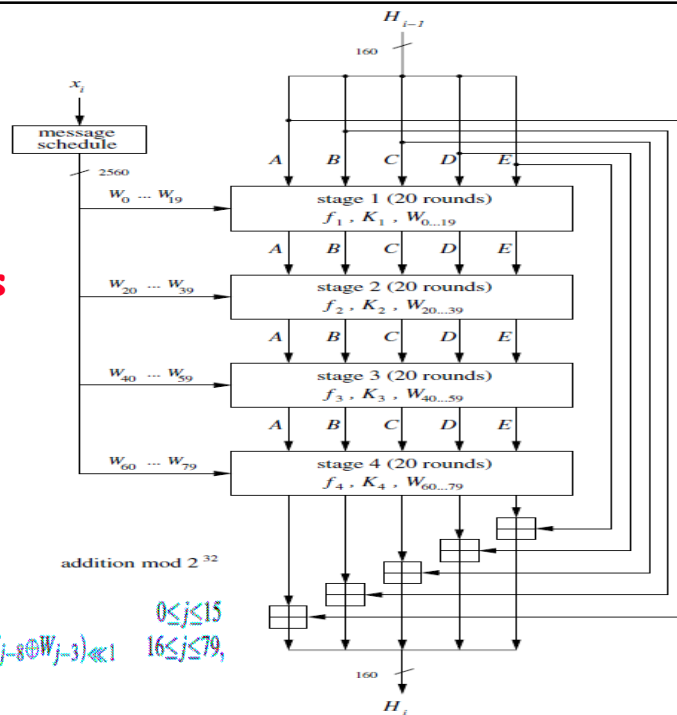
ECE597/697 Koren Part.11 .17

Adapted from Paar & Pelzl, "Understanding Cryptography," and

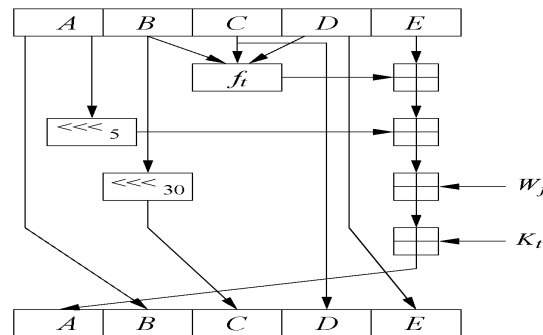
SHA-1: four stages

$$W_j = \begin{cases} x_i^{(j)} & 0 \leq j \leq 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3}) \ll 1 & 16 \leq j \leq 79 \end{cases}$$

ECE597/697 Koren Part.11 .18



SHA-1: Internals of a Round



$$A, B, C, D, E = (E + f_t(B, C, D) + (A) \lll 5 + W_j + K_t), A, (B) \lll 30, C, D$$

Stage t	Round j	Constant K_t	Function f_t
1	0 ... 19	$K = 5A827999$	$f(B, C, D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20 ... 39	$K = 6ED9EBA1$	$f(B, C, D) = B \oplus C \oplus D$
3	40 ... 59	$K = 8F1BBCDC$	$f(B, C, D) = (B \oplus C) \vee (B \oplus D) \vee (C \oplus D)$
4	60 ... 79	$K = CA62C1D6$	$f(B, C, D) = B \oplus C \oplus D$

ECE597/697 Koren Part.11 .19

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Predicted Demise of SHA-1

- ◆ In 2012 it was predicted that 2^{61} attempts will be needed to find a collision in SHA-1
- ◆ It was then predicted (based on Moore's law) that in 2017-2018 such an attack will be possible to mount at a price of about \$200K using GPUs
- ◆ SHA-2 (a greatly modified SHA-1) was released in 2001 and has several versions: SHA-224, SHA-256, SHA-384 and SHA-512
- ◆ In 2015 several attacks have been shown to be feasible on reduced complexity SHA-2 versions

ECE597/697 Koren Part.11 .20

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-3: History

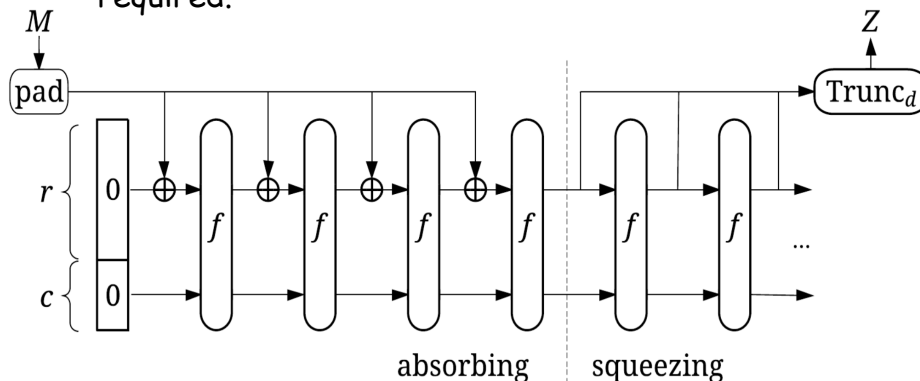
- ◆ Competition started in November 2007 (by NIST)
- ◆ Requirements: Output hash values of 224, 256, 384, 512 bits (> 160)
- ◆ Similar process as the AES competition
- ◆ Maximum message length at least $2^{64} - 1$ bits
- ◆ Implementable in a wide range of hardware and software platforms
- ◆ Deadline for first round: October 2008
- ◆ 14 2nd round candidates announced: July 2009
- ◆ 5 finalists: Dec. 2010
- ◆ In October 2012: winner announced: Keccak by G. Bertoni, J. Daemen, M. Peeters, G. Van Assche
- ◆ August 2015; official publication by NIST of SHA-3

ECE597/697 Koren Part.11 .21

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-3 (Keccak - 2012)

- ◆ Based on a SPONGE function construction
- ◆ Absorbing phase: read in the padded input bit string
- ◆ Squeezing phase: output as many hash bits as required.



ECE597/697 Koren Part.11 .22

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-3 (2)

- ◆ Sponge transformation function $f : \{0, 1\}^b \rightarrow \{0, 1\}^b$ operates on a state S of b bits
- ◆ The state S splits into: $S = R || C$
 - R is of bit length r (r is the bit rate)
 - C is of bit length c (c is the capacity)
- ◆ Input M is divided in blocks M_i of r bits and padded to a multiple of r : $M = M_1 || M_2 || \dots || M_t$ (padding 10 ^{ℓ})
- ◆ A sponge round updates the state $S \leftarrow f(R \oplus M_i || C)$
- ◆ S initialized to 0
- ◆ Capacity part C is not directly linked to the input during the absorbing phase or the output during the squeezing phase
- ◆ Capacity c is linked to the resistance level against a collision attack - can stand an attack up to complexity $2^{c/2}$
- ◆ r affects run time efficiency, c controls security

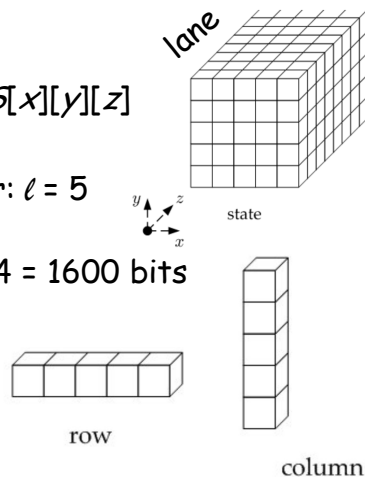
ECE597/697 Koren Part.11 .23

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-3 (3)

- ◆ State is a 3-dimensional array: $S[x][y][z]$ has $b = 5 \times 5 \times 2^\ell$ bits $0 \leq \ell \leq 6$
- ◆ 2^ℓ word length: 32-bit processor: $\ell = 5$ yields a $25 \cdot 32 = 800$ bit state
- ◆ 64-bit processor: $\ell = 6 \Rightarrow 25 \cdot 64 = 1600$ bits
- ◆ SHA-3 is Keccak- $f[1600]$
- ◆ Number of rounds: $12 + 2\ell$
- ◆ round comprises 5 steps:

$$S \leftarrow \iota \circ \chi \circ \pi \circ \rho \circ \theta(S)$$
- ◆ θ : replace a bit by XOR of this bit and the parity of two neighboring columns - provides minor diffusion
- ◆ ρ : rotate the bits in a lane
- ◆ π : permute lanes (permutation in the (x, y) -plane)

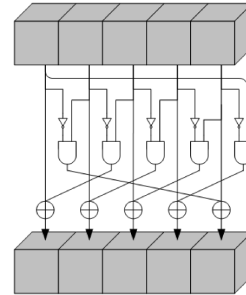


ECE597/697 Koren Part.11 .24

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-3 (4)

- ♦ χ : replace a bit by combining it with 2 subsequent bits in a row
- ♦ Flip bit if neighbors exhibit 01 pattern - non-linear step
- ♦ ι : add a round constant to lane (break symmetry)
- ♦ SHA-3: $r = 1088$ and $c = 512$
permutation width: 1600
security strength 256



ECE597/697 Koren Part.11 .25

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

SHA-3: security

Function	Output Size	Security Strength	
		Collision	Preimage
SHA-1	160	< 80	160
SHA-224	224	112	224
SHA-512/224	224	112	224
SHA-256	256	128	256
SHA-512/256	256	128	256
SHA-384	384	192	384
SHA-512	512	256	512
SHA3-224	224	112	224
SHA3-256	256	128	256
SHA3-384	384	192	384
SHA3-512	512	256	512

ECE597/697 Koren Part.11 .26

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Lessons Learned

- Hash functions are keyless. The two most important applications are: digital signatures and in message authentication codes such as HMAC
- The 3 security requirements for hash functions are one-wayness, second preimage resistance and collision resistance
- Hash functions should have at least 160-bit output length in order to withstand collision attacks; 256 bit or more is desirable for long-term security
- Some security weaknesses have been found in SHA-1, and it is being phased out. The SHA-2 algorithms appear to be more secure but also start to be questionable
- The SHA-3 competition resulted in new standardized hash functions