

# Cryptographic Hashing at Crossroads

Josef Pieprzyk

Centre for Advanced Computing – Algorithms and Cryptography,  
Department of Computing, Macquarie University  
Sydney, AUSTRALIA

EFPE - June, 2008

## Outline

- 1 Overview
- 2 Introduction
- 3 Hashing based on Block Ciphers
- 4 Custom Designed Hash Functions
- 5 Hashing Based on Intractable Problems
- 6 NIST Call for SHA-3

## Talk overview

- Introduction
  - Properties of hash functions
  - Structures for hash functions
  - Generic attacks on hash functions
- Hashing based on block ciphers
  - First constructions
  - Double-block hashing
  - Pros and cons
- Custom-designed hash functions
  - MD family
  - Fork-256
  - Message expansion
  - Pros and cons
- Hashing based on intractable problems
  - MASH-1, RSA, DL
  - Knapsack, Tillich-Zémor, LASH-160, VSH, SWIFFT
  - Pros and cons
- NIST call for SHA-3

## Outline

- 1 Overview
- 2 Introduction**
- 3 Hashing based on Block Ciphers
- 4 Custom Designed Hash Functions
- 5 Hashing Based on Intractable Problems
- 6 NIST Call for SHA-3

## Classical Hashing

In Computer Science hashing have been used for searching.

- Hash functions defined as

$$H : \{0, 1\}^N \rightarrow \{0, 1\}^n$$

where  $N$  is the length of the key record and  $n$  is the length of indices that point to appropriate **buckets**, where  $N > n$ .

- Expected properties
  - 1 uniform distribution of **collisions** – buckets contain roughly the same number of records,
  - 2 efficient computation of indices.

## Cryptographic Hashing - Definition

Hash function accepts inputs of an arbitrary length and produces a **message digest** of a fixed length  $n$  bits, i.e.

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

Hash function should be:

- 1 **preimage resistant**, i.e. knowing a digest, it is intractable to find any message that hashes to the digest in hand,
- 2 **2nd preimage resistant**, i.e. given a message  $m$  and its digest  $d = H(m)$ , it should be intractable to find another message

$$m' \text{ such that } H(m') = H(m) = d$$

- 3 **collision resistant**, i.e. given the hash function, its is intractable to find any pair of messages  $(m, m')$  such that

$$H(m') = H(m)$$

## Properties of Hash Functions

Hash functions are also expected to provide:

- **Pseudorandomness** – security of HMAC is based on the assumption that keyed compression function is a pseudorandom function (PRF) family.
- **Random oracle** – security of RSA-OAEP and RSA-PSS relies on the assumption that hash functions are random oracles.
- **Indistinguishability** – NIST's draft call requires the output of hash function to be indistinguishable from a random oracle.

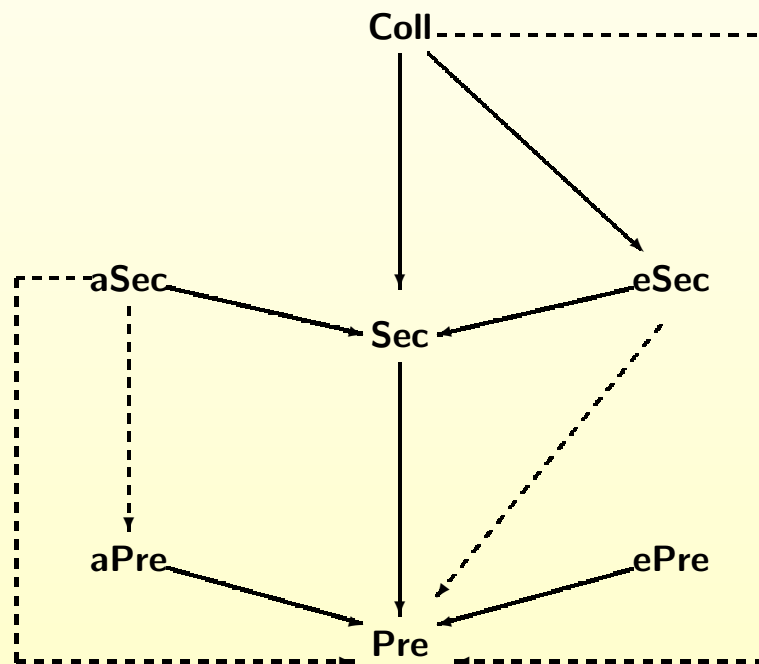
## Family of Hash Functions

- Definition

$$H : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^n$$

- Properties (Rogaway and Shrimpton)
  - Preimage (**Pre**) resistant; Second preimage (**Sec**) resistant; and Collision (**Coll**) resistant
  - everywhere Preimage resistant (**ePre**) – random key and fixed challenge message
  - always Preimage resistant (**aPre**) – fixed key and random challenge message
  - everywhere Second preimage resistant (**eSec**)
  - always Second preimage resistant (**aSec**)

## Relation Among Properties



## Structures for Hash Functions

How to build  $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$  ?

- **Merkle-Damgård construction** – sequential application of compression function

$$h : \{0, 1\}^N \rightarrow \{0, 1\}^n.$$

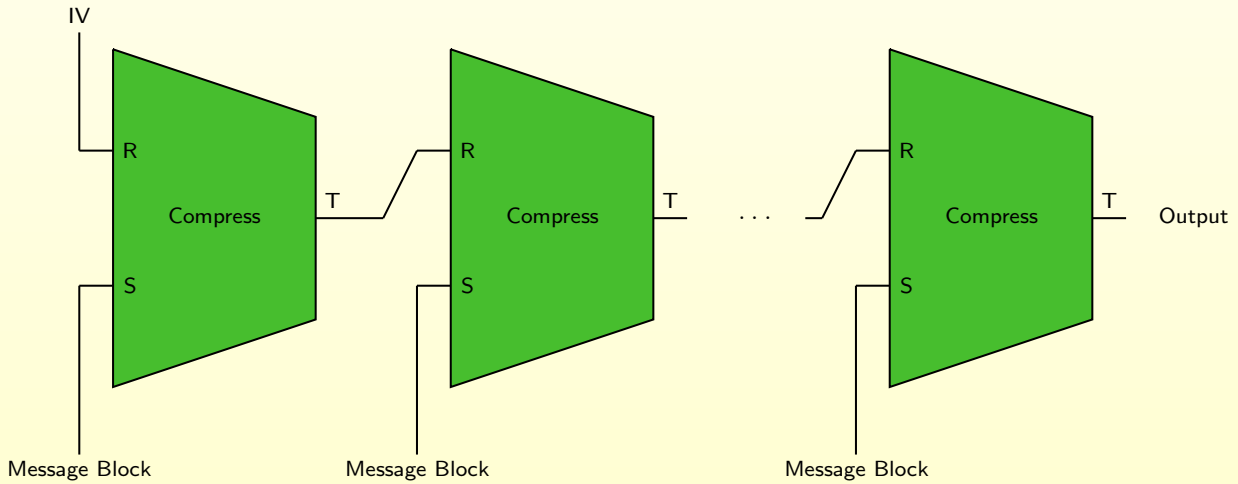
- **Damgård construction** – tree of the compression function  
 $h : \{0, 1\}^N \rightarrow \{0, 1\}^n.$
- **Sponge construction** – application of permutation

$$f : \{0, 1\}^N \rightarrow \{0, 1\}^N$$

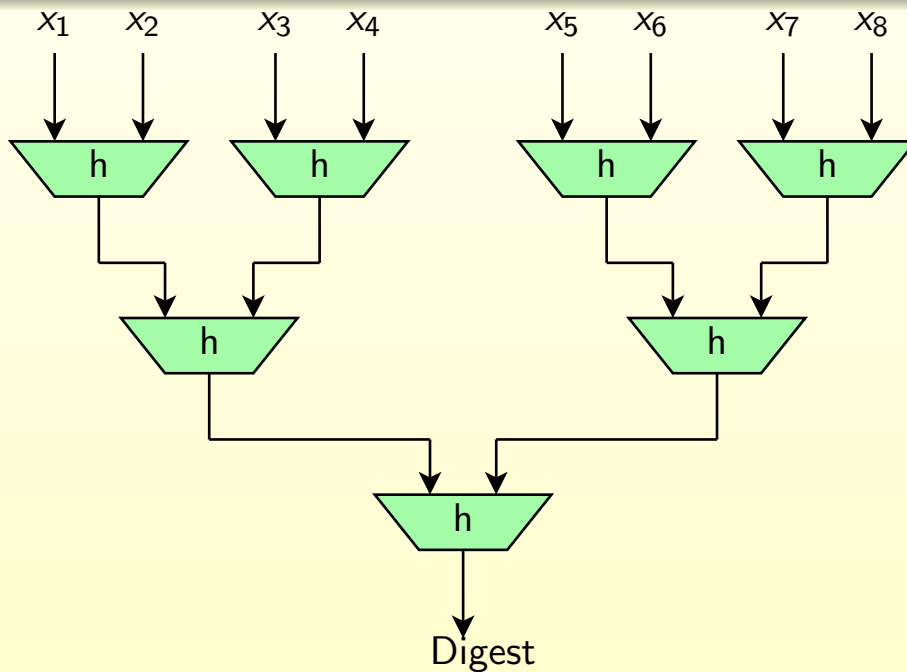
and two phases: absorbing messages and squeezing out digest.

- **Permutation and checksums** (MD2)
- **Wide-pipes and double-pipes**

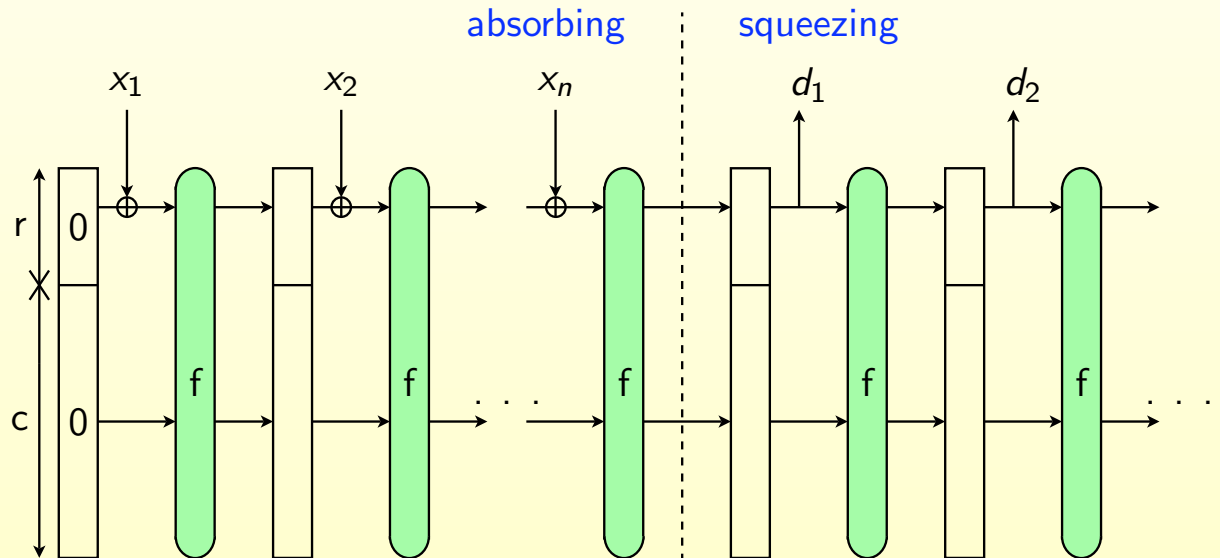
## Merkle-Damgård Structure



## Damgård Tree Construction

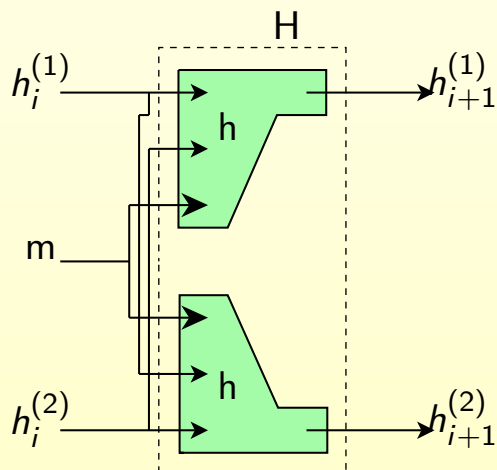


# Sponge Construction



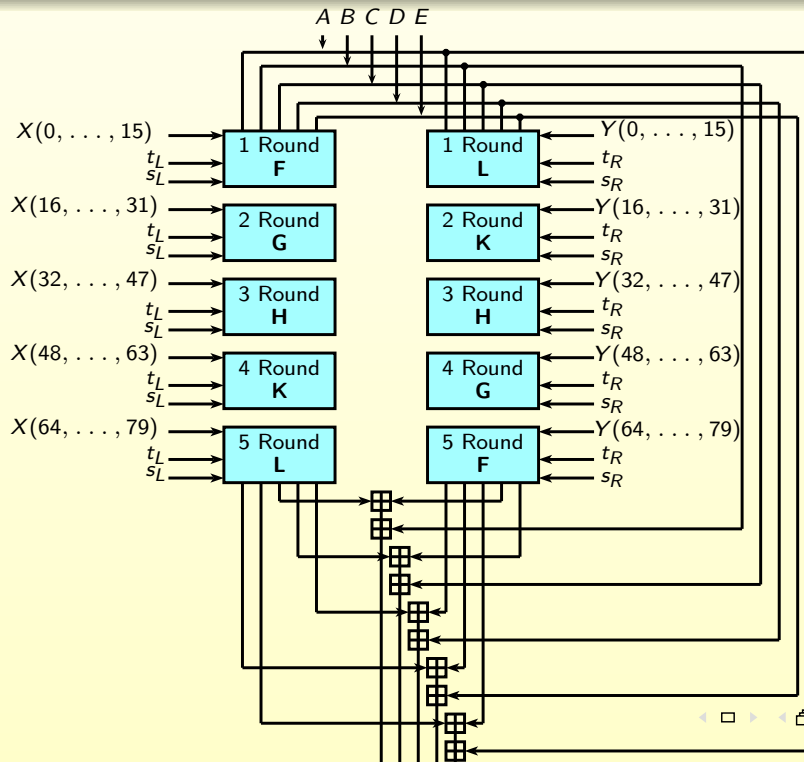
# Double-pipe

$$H(h_i^{(1)}, h_i^{(2)}, m) = (h(h_i^{(1)}, h_i^{(2)}, m), h(h_i^{(2)}, h_i^{(1)}, m))$$

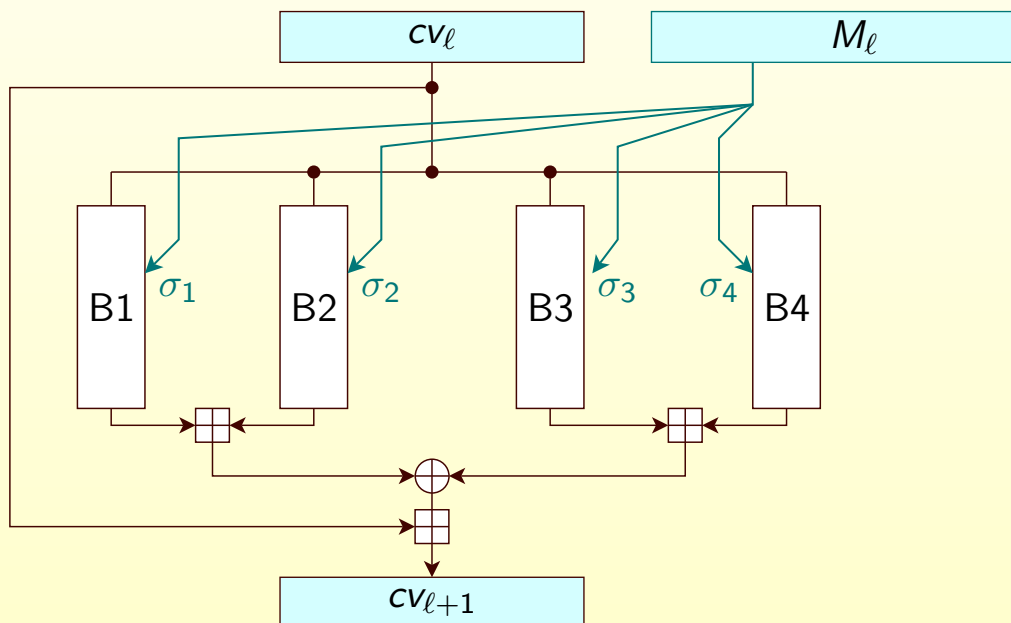




## Double-block Hash - Merkle (1989)



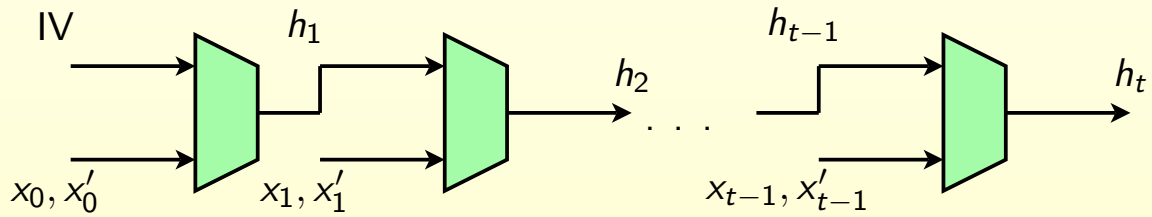
## Four Parallel Chains - FORK





## Multicollisions

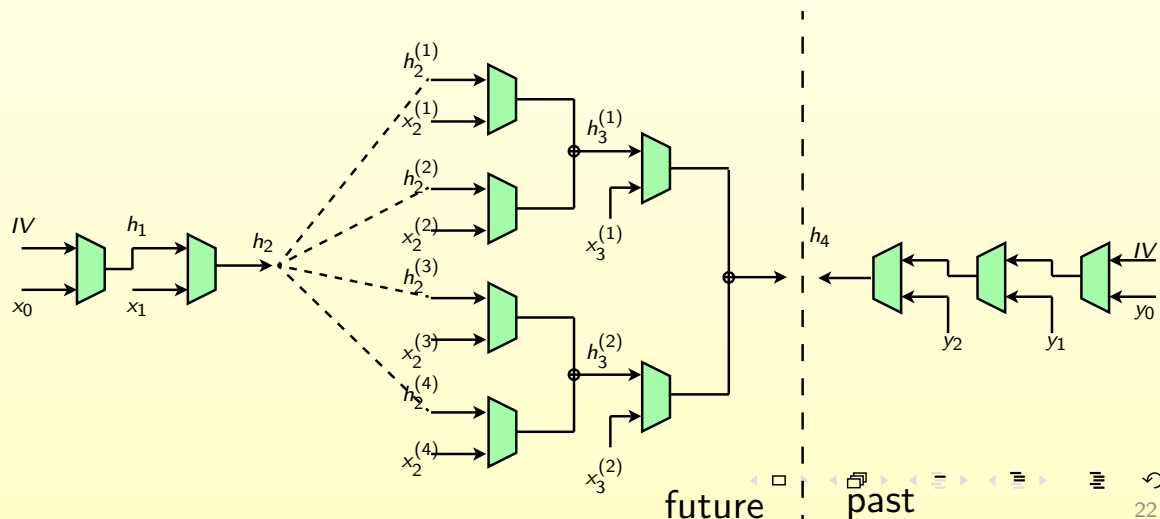
Assume we have  $t$  collisions



then we can create  $2^t$  multicollisions.

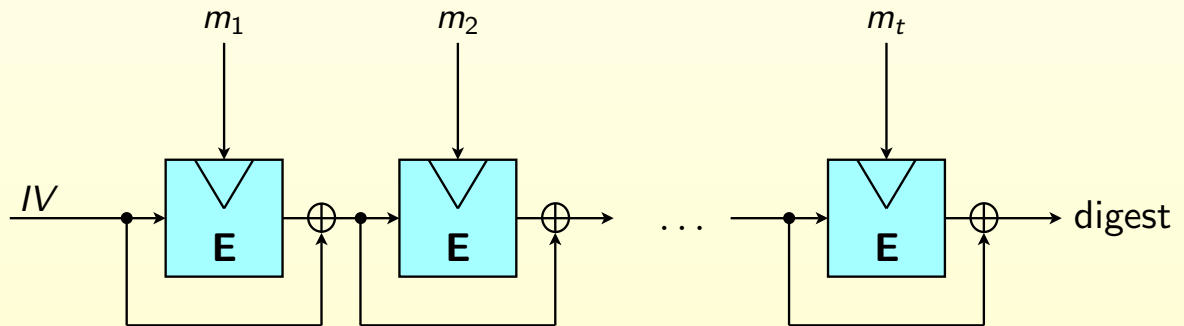
## Herding Attack

Finding collision for a given message ( $IV \parallel y_0 \parallel y_1 \parallel y_2$ ). Create a collision tree of  $2^t$  values costs  $O(2^{t/2} 2^{n/2})$ . Finding the match for  $h_2$  takes  $O(2^{n-t})$ .

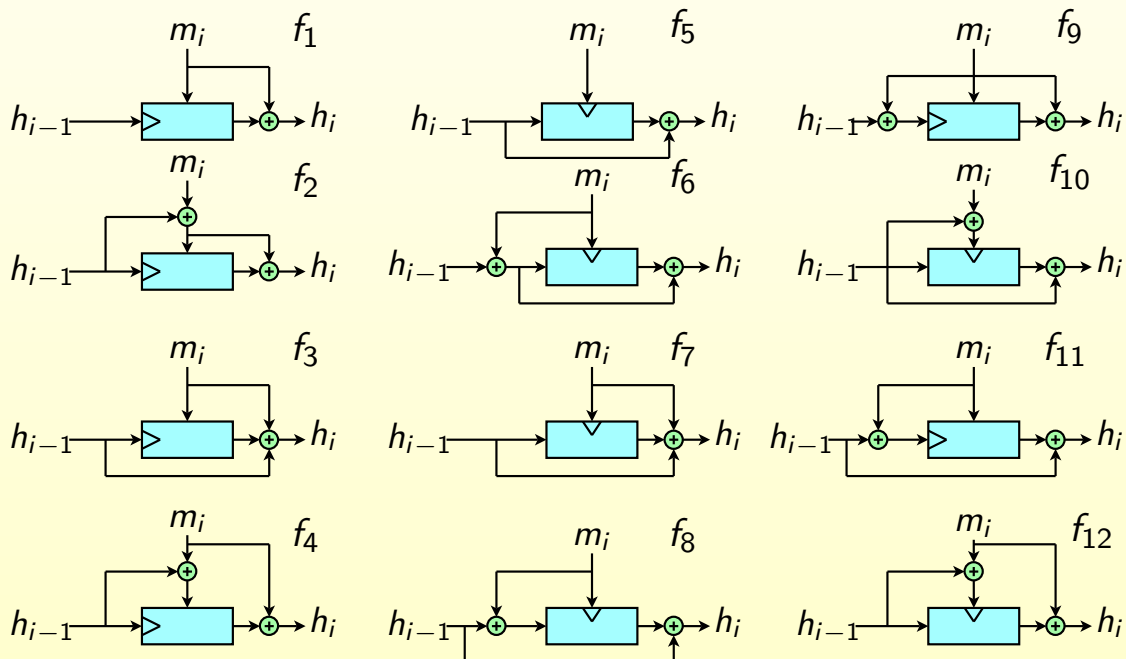




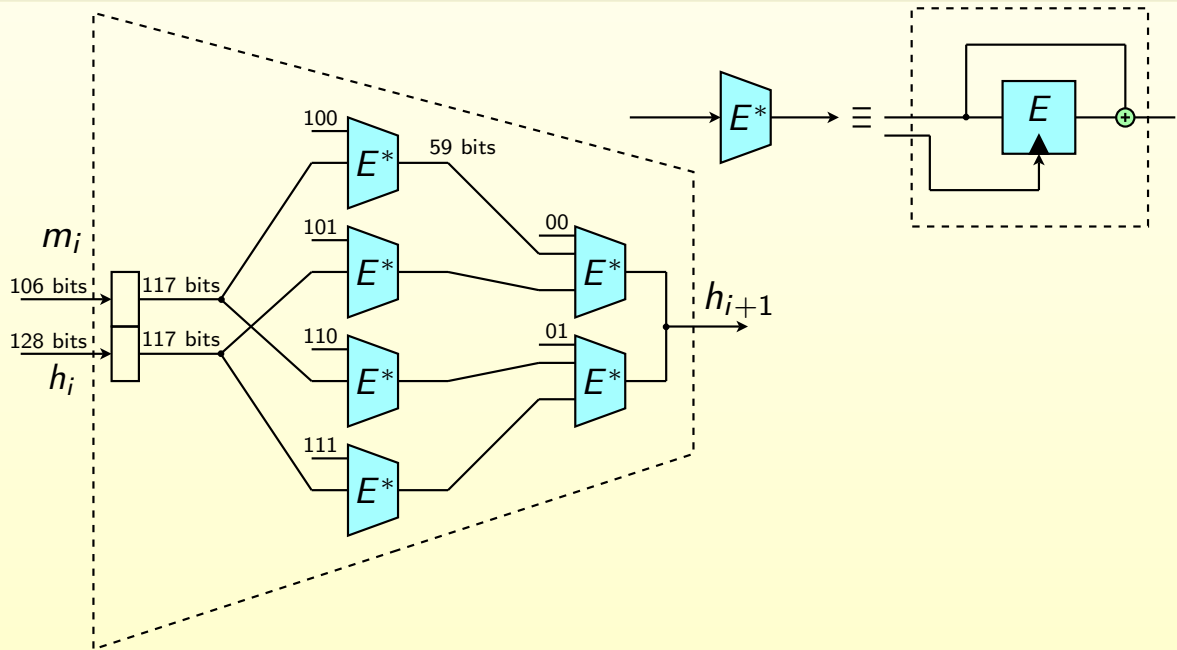
## First Constructions - Davies-Meyer



## MD Hashing from Block Ciphers - Secure Constructions

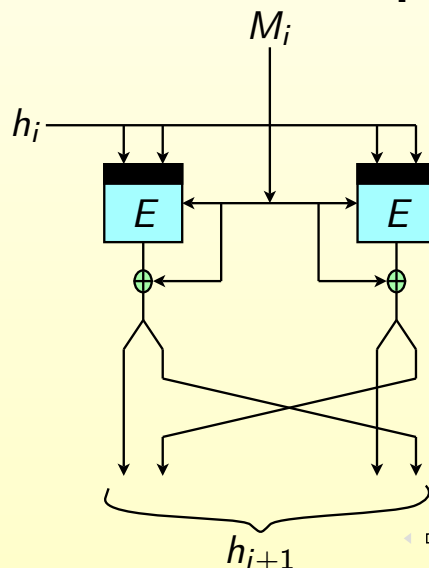


## Double-block Hash - Merkle (1989)



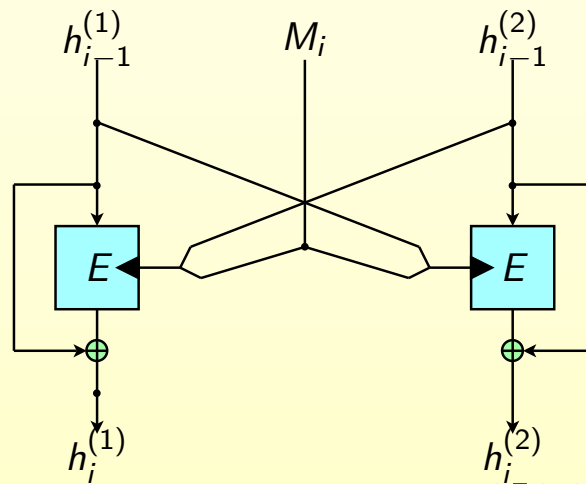
## Double-block Hash - MDC-2

- Designed by Brachtel et al in 1990 (ANSI X9.31 standard)
- Proof that when the block size is 128-bits, than the collision attack requires more than  $2^{74.9}$  queries [Steinberger 2007]



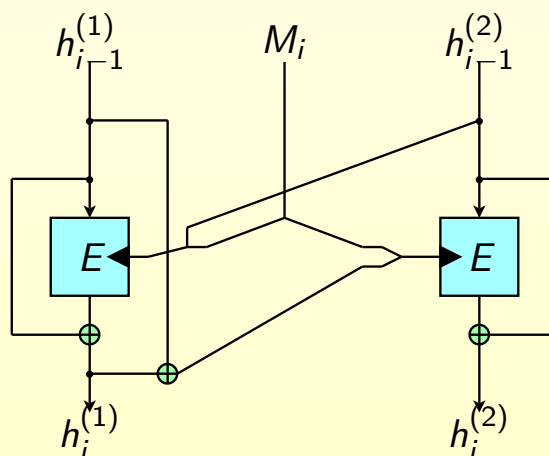
## Abreast-DM

- Designed by Lai and Massey in 1990
- Underlying block cipher is IDEA (but works for any cipher with key and message sizes  $2n$  and  $n$ , respectively)



## Tandem-DM

- Designed by Lai and Massey in 1990
- Underlying block cipher is IDEA (but works for any cipher with key and message sizes  $2n$  and  $n$ , respectively)



## Hashing Based on Block Ciphers - Pros and Cons

### PROS

- Block cipher can be easily adapted for hashing
- Reuse of existing and well developed technology
- Block cipher designs withstood the test of time (DES)
- 

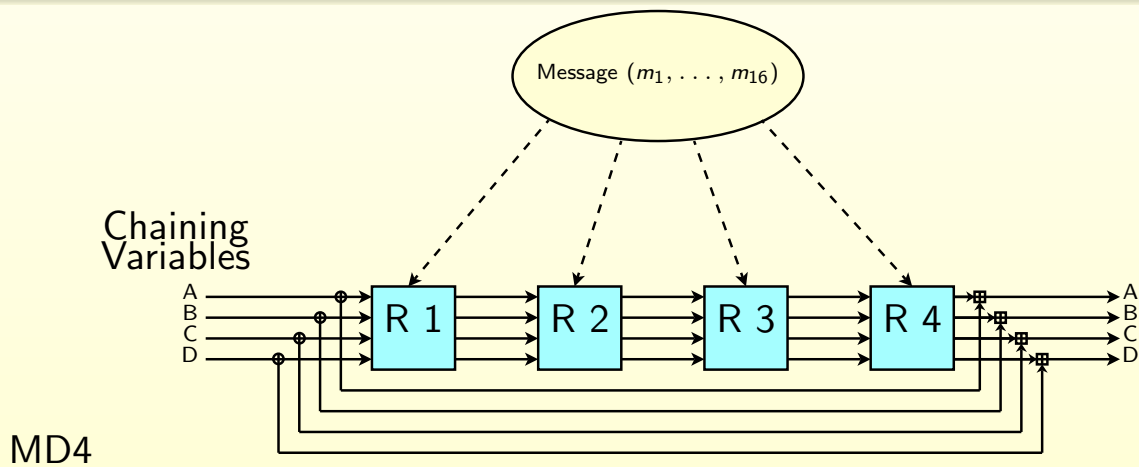
### CONS

- mismatch between key scheduling (cipher) and message expansion (hash)
- hashing is slower than custom-designed one

## Outline

- 1 Overview
- 2 Introduction
- 3 Hashing based on Block Ciphers
- 4 Custom Designed Hash Functions
- 5 Hashing Based on Intractable Problems
- 6 NIST Call for SHA-3

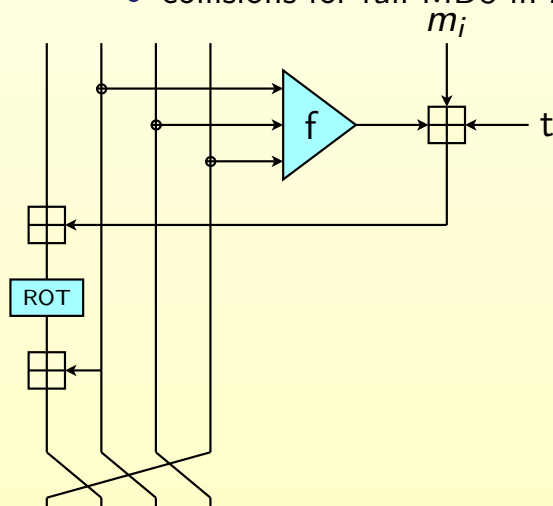
## MD Family



- Designed by Rivest in 1990
- Based on Feistel permutations (3 rounds/48 steps)
- collision resistance:
  - collisions for 2 rounds [Merkle90, denBoerBosselaers91]
  - collisions for full MD4 by hand [Wang04]

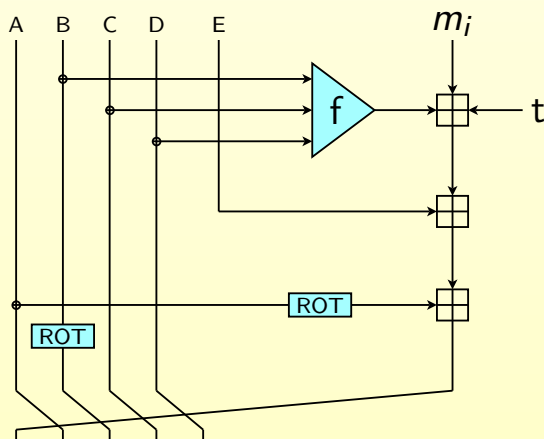
## MD5

- Designed by Rivest in 1991
- Based on Feistel permutations (4 rounds/64 steps)
- collision resistance:
  - collisions for compression function [Dobbertin96]
  - collisions for full MD5 in  $2^{39}$  [Wang04]



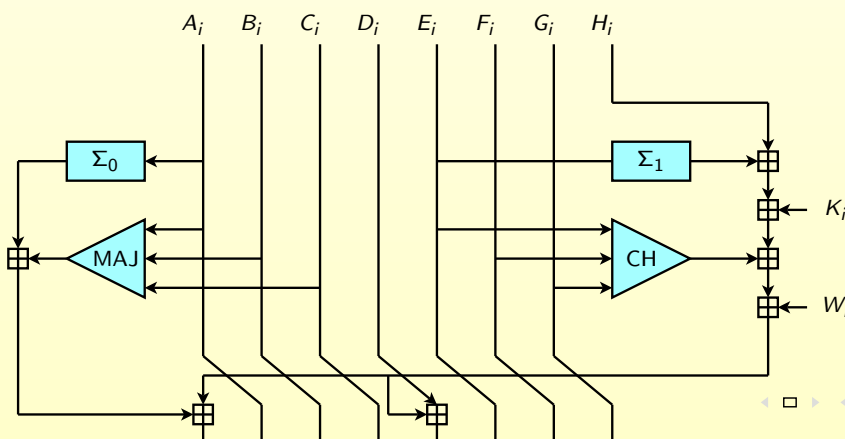
## SHA-1

- SHA-0 designed by NIST in 1993 and in 1995 upgraded to SHA-1
- Based on Feistel permutations (5 rounds/80 steps)
- collision resistance:
  - collisions for SHA-0 in  $2^{51}$  [Joux04]
  - collisions for full SHA-1 in  $2^{63}$  [Wang05]

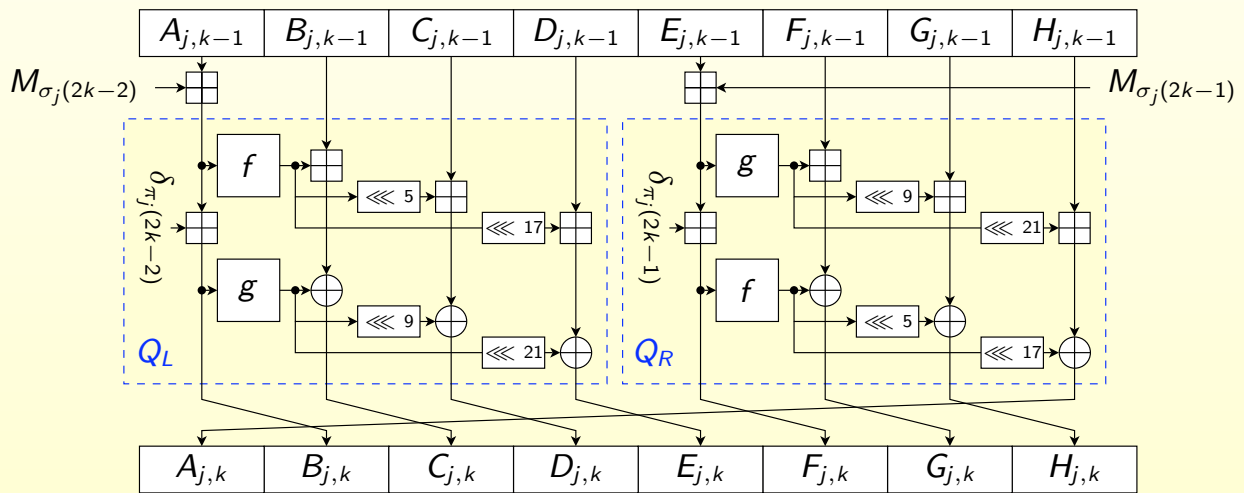


## SHA-256

- NIST published SHA-256 in 2001 (FIPS PUB180-2)
- Feistel permutation (64 steps)
- collision resistance:
  - collisions for SHA-256<sub>23</sub> with complexity  $2^{18}$
  - collisions for SHA-256<sub>24</sub> with complexity  $2^{28.5}$  [Indestege et al, March 2008]



# FORK256

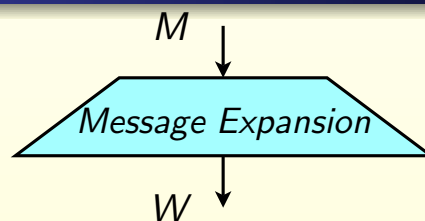


where

$$f(x) = x + (ROTL^7(x) \oplus ROTL^{22}(x))$$

$$g(x) = x \oplus (ROT^{13}(x) + ROT^{27}(x))$$

# Message Expansion



- round permutations (MD4, MD5, RIPE-MD, HAVAL, FORK-256)
- linear code (SHA-0, SHA-1). For SHA-1

$$W_t = \begin{cases} M_t & \text{for } 0 \leq t \leq 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & \text{for } 16 \leq t \leq 79 \end{cases}$$

- nonlinear code. For SHA-256

$$W_t = \begin{cases} M_t & \text{for } 0 \leq t \leq 15 \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & \text{for } 16 \leq t \leq 63 \end{cases}$$

where  $\sigma_0(x) = ROTL^7(x) \oplus ROTL^8(x) \oplus SHR^3(x)$  and  $\sigma_1(x) = ROTL^{17}(x) \oplus ROTL^{19}(x) \oplus SHR^{10}(x)$

## Custom Designed Hash - Pros and Cons

### PROS

- Specifically designed for hashing
- Diffusion as the main cryptographic property
- S-box theory comes handy
- Basic operations can be chosen to maximize speed
- Two basic components: message expansion and processing of chain variables

### CONS

- Security is not proven
- Hashing is a relatively new comer and lagging behind secret-key cryptography

## Outline

- 1 Overview
- 2 Introduction
- 3 Hashing based on Block Ciphers
- 4 Custom Designed Hash Functions
- 5 Hashing Based on Intractable Problems**
- 6 NIST Call for SHA-3

## Modular Arithmetic Secure Hash – MASH-1

**Authors:** ISO/IEC SC27, Girault and Misarsky

**Assumption:** Square root modulo  $N$

**Construction:** Compression function is

$$H_i = ((m_i \oplus H_{i-1})^\wedge)^2 \bmod N \oplus H_{i-1}$$

Size of of the digest is  $\frac{n}{2}$ .

- Security:**
- Best preimage attack  $2^{n/2}$
  - Best collision attack  $2^{n/4}$
  - No security reduction proof

## RSA-based Hash

**Authors:** Gibson

**Assumption:** Discrete logarithm and factoring

**Construction:** Hashing is done for the whole message  $M$

$$H(M) = g^M \pmod{N}$$

where  $g$  is a generator of the cyclic group  $\mathcal{Z}_N^*$  and  $N = pq$ ,  $p$  and  $q$  are strong primes.

**Security:** Finding collisions reveal the factors of  $N$ .

## Discrete Logarithm Hash

**Authors:** Chaum, van Heijst and Pfitzmann

**Assumption:** Discrete logarithm

**Construction:** Hashing is done for the whole message

$$M = (m_1, m_2)$$

$$H(M) = g_1^{m_1} g_2^{m_2} \pmod{p}$$

where  $p$  is prime and  $g_1, g_2$  are two randomly chosen generators of  $\mathcal{Z}_p^*$

**Security:** Finding a collision solves the discrete logarithm

$$g_2 = g_1^\alpha$$

## Knapsack Hash

**Authors:** Impagliazzo and Naor

**Assumption:** Knapsack

**Construction:** Compression function  $H : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$

$$H(M) = \sum_{i=1}^n m_i a_i \pmod{2^\ell}$$

where  $M = (m_1, \dots, m_n)$  is a message and  $A = (a_1, \dots, a_n)$  is a collection of randomly chosen elements ( $a_i \leq 2^\ell$ )

**Security:** there is a reduction proof

## Tillich-Zémor Hash

**Authors:** Tillich and Zémor

**Assumption:** Hardness of finding a short factorization in  $SL(2, 2^n)$

**Construction:** Hash function operates on arbitrary long messages  $M = (m_1, \dots, m_r)$  and returns an element of  $SL(2, 2^n)$ .

- Given two public elements

$$A = \begin{bmatrix} x & 1 \\ 1 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} x & x+1 \\ 1 & 1 \end{bmatrix}.$$

## Tillich-Zémor Hash

- The message is first converted into the corresponding sequence of  $A$ 's and  $B$ 's according to the function

$$\pi(m_i) = \begin{cases} A & \text{if } m_i = 0 \\ B & \text{otherwise} \end{cases}$$

- Digest is  $H(M) = \pi(b_1)\pi(b_2)\dots\pi(b_r)$

**Security:** There is an evidence that the hash is not collision free for some parameters.

## LASH-160

**Authors:** Bentahar, Page, Saarinen, Silverman, and Smart

**Assumption:** Knapsack (matrix multiplication)

**Construction:** Compression function

$$H : \{0, 1\}^{320} \times \{0, 1\}^{320} \rightarrow \{0, 1\}^{320}$$

$$\begin{array}{c}
 \text{40 bytes} \\
 \left\{ \left[ \begin{array}{c} R \\ \vdots \\ R \end{array} \right] \oplus \left[ \begin{array}{c} S \\ \vdots \\ S \end{array} \right] + \left[ \begin{array}{c} \text{640 columns} \\ \vdots \\ H \\ \vdots \end{array} \right] \right\} \cdot \begin{array}{c} \text{bit vec} \\ \left[ \begin{array}{c} r_0 \\ \vdots \\ r_{319} \\ \hline s_0 \\ \vdots \\ s_{319} \end{array} \right] \end{array}
 \end{array}$$

where  $r_0, \dots, r_{319}$  and  $s_0, \dots, s_{319}$  are bit expansions of  $R$  and  $S$ , respectively.

**Security:** Collisions in  $2^{(4/11)160}$  and preimages in  $2^{(4/7)160}$

## Very Smooth Hash – VSH

**Authors:** Contini, Lenstra and Steinfeld

**Assumption:** Very smooth number nontrivial modular square root (VSSR)

**Construction:** Compression function is

$$h_{i+1} = H(m, h_i) = h_i^2 \times \prod_{j=1}^k p_j^{m_j} \pmod{N}$$

where the message block  $m = (m_1, \dots, m_k)$  has  $k$  bits, primes  $p_1, \dots, p_k$  are smooth and  $N$  is an RSA modulus

**Security:** Finding collisions is as hard as solving VSSR

## SWIFFT Hash

**Authors:** Lyubashevsky, Micciancio, Peikert and Rosen

**Assumption:** Worst-case hardness in cyclic/ideal lattices

**Construction:** Computations are done in  $\mathcal{R} = \mathbb{Z}_p[\alpha]/(\alpha^n+1)$ , where  $p$  is a prime

- Message block  $X = (x_1, \dots, x_m)$  and each  $x_i \in \mathcal{R}$  (contains  $mn$  bits)
- Fix matrix  $A = (a_1 \dots, a_m)$  is chosen at random with  $a_i \in \mathcal{R}$
- digest  $H_A(X) = \sum_{i=1}^m a_i \cdot x_i \in \mathcal{R}$  of the length  $m \cdot \log_2 p$
- instantiation for  $m = 16$ ,  $n = 64$ , and  $p = 257$  and  $A$  – compressing 1024 bits to 528 bits.

**Security:** Collision resistant for random matrices  $A$

## Hashing Based on Intractable Problems - Pros and Cons

### PROS

- Security properties can be proven rigorously
- Security reduction can be constructive giving lower bounds on possible attacks
- recent constructions such as VSH and SWIFFT are closing the efficiency gap

### CONS

- Efficiency is still a problem

## Outline

- 1 Overview
- 2 Introduction
- 3 Hashing based on Block Ciphers
- 4 Custom Designed Hash Functions
- 5 Hashing Based on Intractable Problems
- 6 NIST Call for SHA-3

## SHA-3 – Background

- Impact of Xiaoyun Wang attacks - On April 26, 2006, NIST comments:  
“NIST accepts that Prof. Wang has indeed found a practical collision attack on SHA-1.”
- January 23, 2007 – NIST announces the Development of New Hash Algorithm(s), Secure Hash Standard
- November 2, 2007 – NIST calls for SHA-3 submissions
- October 31, 2008 – deadline for submissions

details see

<http://csrc.nist.gov/groups/ST/hash/index.html>

## SHA-3 – Requirements

New SHA-3 algorithm should support

- digital signature standard (DSS) – FIPS 186-2
- keyed-hash message authentication code (HMAC) – FIPS 198
- pair-wise key establishment – SP 800-56A
- random number generator – SP 800-90

Terms of reference

- SHA-3 algorithms should offer features that exceed the SHA-2 hash functions
- NIST expects SHA-3 to have a security strength that is at least as good as SHA-2
- NIST *strongly* desires a single hash algorithm family
- However, if more than one suitable candidate is identified, NIST may consider more than one family for inclusion

## SHA-3 – Requirements - cont.

- algorithms with tunable security parameters
- NIST encourages constructions that differ from the MD model
- implementable in a wide range of hardware and software platforms
- capable to support digest sizes of 224, 256, 384 and 512 bits
- maximum message length at least  $2^{64} - 1$  bits.

IP issues

- algorithm must be available worldwide and be royalty free

## SHA-3 – Security

- if used as PRF, it must resist any distinguishing attack that requires fewer than  $2^{n/2}$  queries
- collision resistance of  $2^{n/2}$
- preimage resistance of  $2^n$
- second preimage resistance of  $2^{n-k}$  for any message shorter than  $2^k$  bits
- resistance to length-extension attacks
- resistance against other attacks such as multicollision attacks, is encouraged by NIST

## SHA-3 – Evaluation Process

- evaluation process is public
- after the deadline October 31, 2008, NIST will review the submission and organize a public conference
- Round 1 – 12 month review to cull weaker submissions (no modifications). Two workshops at the very beginning and end
- Round 2 – 12-15 months careful evaluation (modification allowed). At the end the third workshop
- Announcement of AHS

## References

- Lars Knudsen, Hash functions and SHA-3, Invited Talk, FSE 2008
- NIST, Cryptographic Hash project,  
<http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>
- Bart Preneel, Hash functions: past, present and future, Invited Talk, Asiacrypt 2005