# Cryptographic hash functions

**Presented by**

Prof. Danilo Gligoroski

Department of Telematics

Faculty of Information Technology, Mathematics and Electrical Engineering

Norwegian University of Science and TechnologyTechnology - NTNU, NORWAY

NTNU
Innovation and Creativity

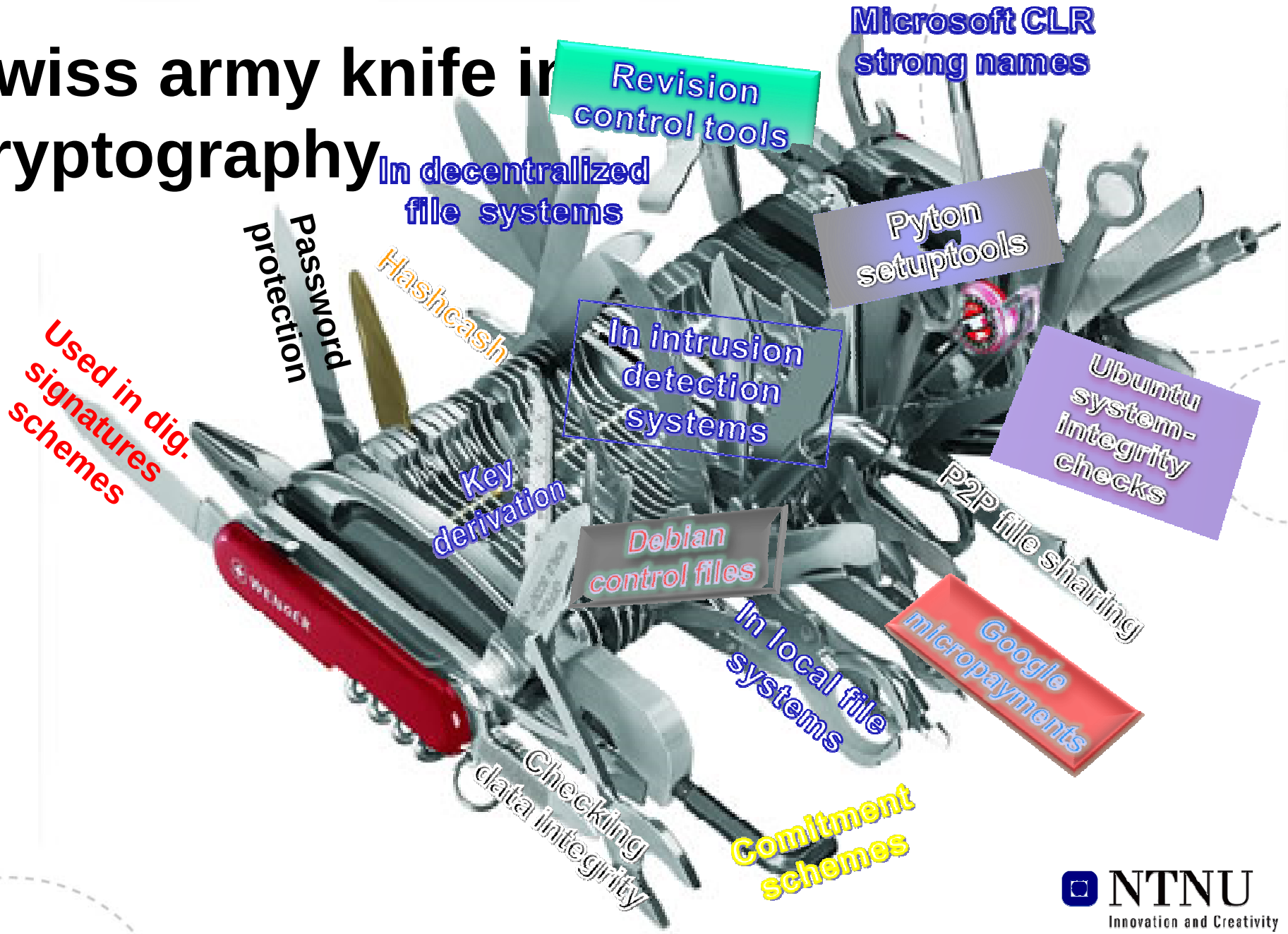
# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- Merkle-Damgård Construction does not act as Random Oracle
- Constructions of iterative hash functions that do not suffer from attacks on MD
- Building blocks in the hash functions
- Description of the main applications of hash functions
- NIST worldwide hash competition for SHA-3
- Advertisment: Blue Midnight Wish Cryptographic Hash Function

# Swiss army knife in cryptography

- One of the most useful mathematical concept that has been invented in the contemporary cryptology in the last 20 years.
- **Main use:**
  - **Part of digital signature schemas**

# Swiss army knife in cryptography




NTNU
Innovation and Creativity

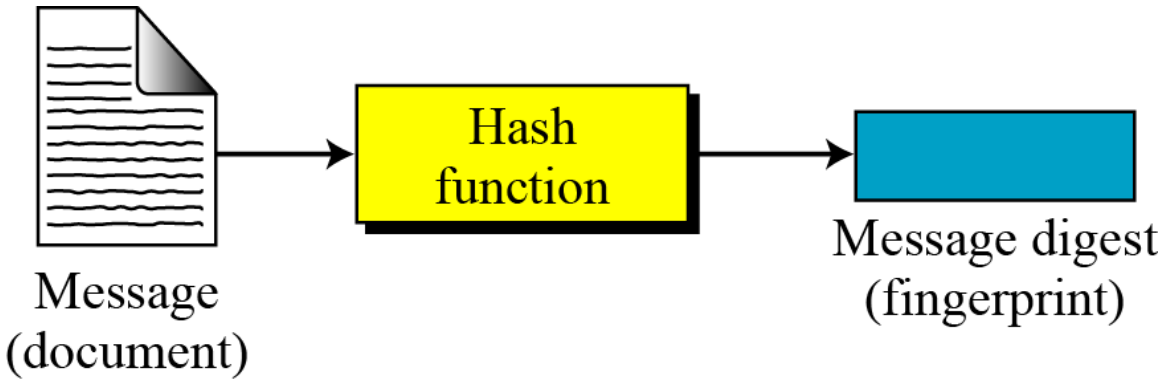
# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- Merkle-Damgård Construction does not act as Random Oracle
- Constructions of iterative hash functions that do not suffer from attacks on MD
- Building blocks in the hash functions
- Description of the main applications of hash functions
- NIST worldwide hash competition for SHA-3
- Advertisment: Blue Midnight Wish Cryptographic Hash Function

# What is a cryptographic hash function?

## Intuitive requirements

- A function that maps a message of an arbitrary length to a n-bit output (the message digest)
  - output to act as a fingerprint
  - if the message digest is transmitted **securely**, then changes to the message to be detected with overwhelming probability




NTNU
Innovation and Creativity

# What is a cryptographic hash function?

## Informal definition

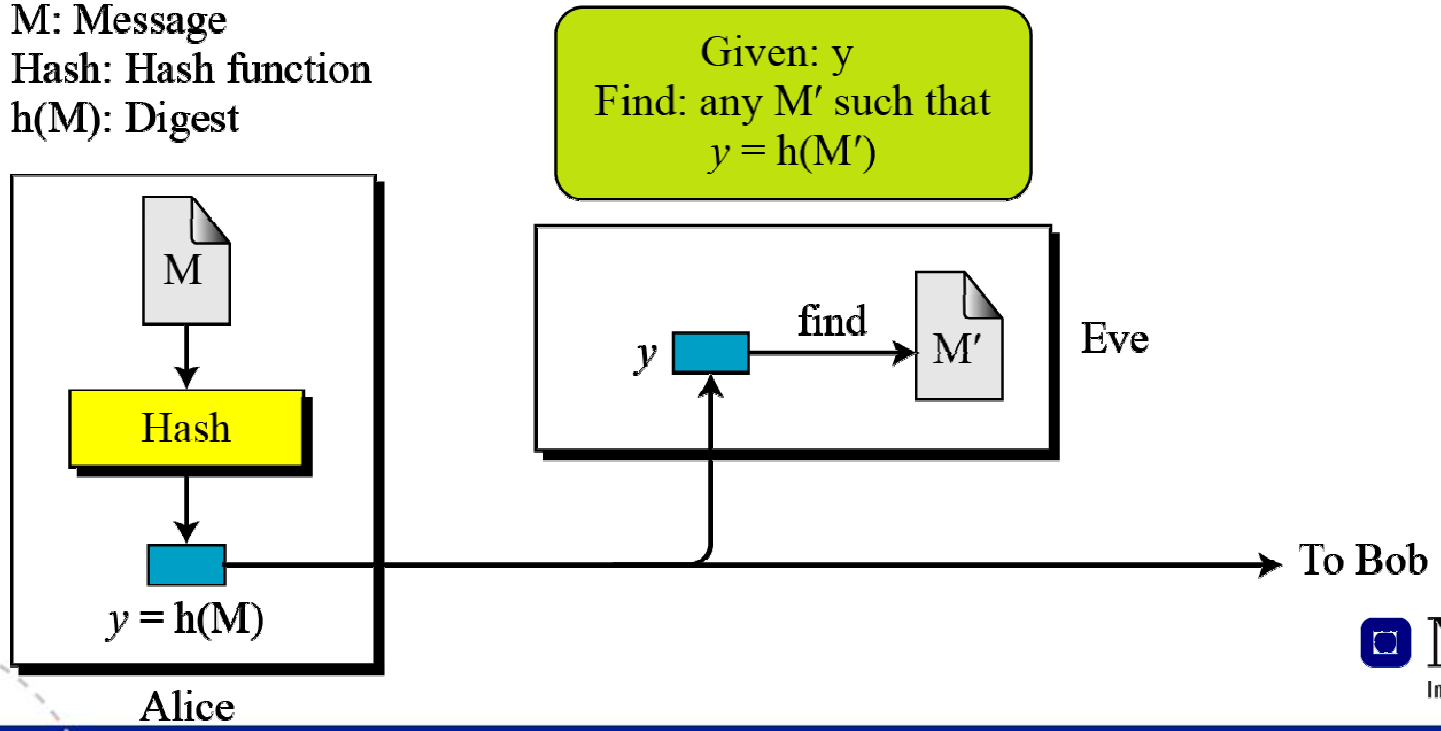
A function $h:\{0,1\}^* \rightarrow \{0,1\}^n$, for a given $n$☠ is called cryptographic hash function if it is

1. **One way**
2. **Collision resistant**

# What is a cryptographic hash function?

## 1. One way

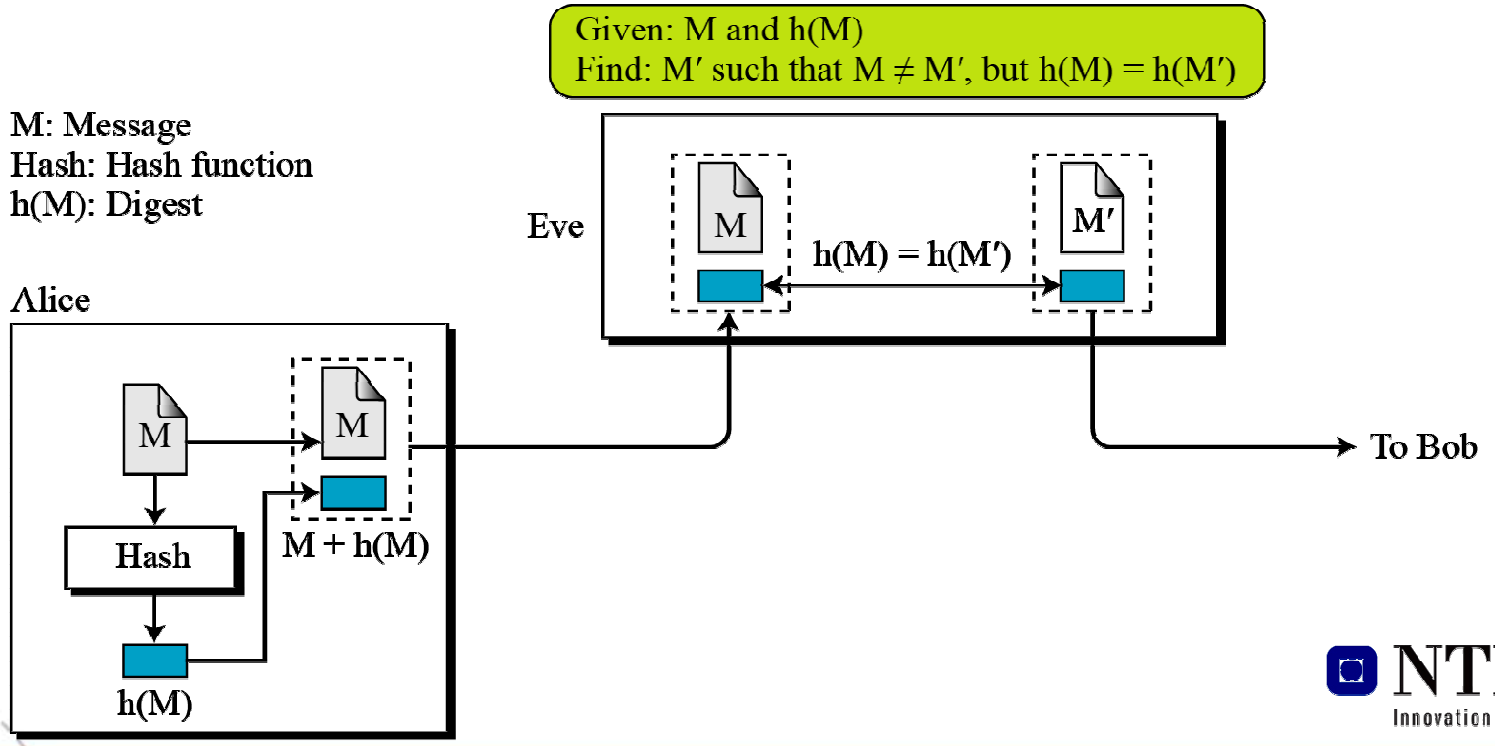
– **Preimage resistant** i.e. it should has the property that it is "easy" to compute $h(M)=y$ for a given $M \in \{0,1\}^*$, but it should be "hard" (or "infeasible") to compute any $M'$ if just the value of $y$ is given.



# What is a cryptographic hash function?

## 1. One way

– **Second preimage resistant** i.e. it should has the property that for a given M it is "easy" to compute $h(M)$, but it should be "hard"' (or "infeasible") to find another $M' \neq M$ such that $h(M')=h(M)$.



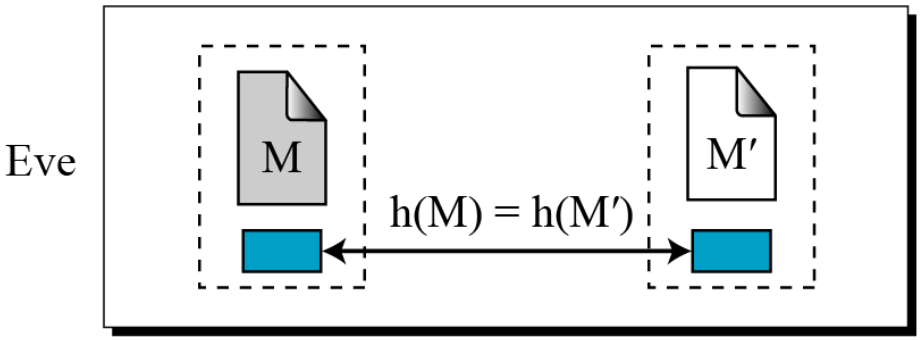
# What is a cryptographic hash function?

## 1. Collision resistant

– It should be “hard” (or “infeasible”) to find two values $M \neq M'$ such that $h(M)=h(M')$.

M: Message
Hash: Hash function
h(M): Digest

Find: M and M′ such that M ≠ M′, but h(M) = h(M′)



# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- **Conflict between theory and practice**
- **Random oracle paradigm**
- **Formal definition of cryptographic hash functions and basic attacks**
- **Choosing the length of hash functions**
- **Merkle-Damgård iterative construction of hash functions**
- **A short history of practical hash functions constructions (and their failures)**
- **Merkle-Damgård Construction does not act as Random Oracle**
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

# Conflicts theory vs. practice

- The first explicit note for the need of one-way functions in cryptography was given by Diffie and Hellman in 1976.
- Several significant theoretical works by Yao in 1982 and Levin in 1985 (connected with even older ideas of by Kolmogorov from mid and late 1960s)
- The existence of one-way functions was connected with the famous question from the complexity theory is P = NP ?

# Conflicts theory vs. practice

- It was shown that if one-way functions exist then $P \neq NP$
- Or vice versa, if P = NP then there are no one-way functions.
- **The first conflict:**
  - It is well known that so far no one has succeeded either to prove or to disprove the claim P = NP,
  - Nevertheless the practical requirements to the designers of the cryptographic hash functions are to construct functions that look like one-way.

# Conflicts theory vs. practice

- Trivial theoretical fact (the pigeonhole principle). Since $h:\{0,1\}^* \rightarrow \{0,1\}^n$, then there are infinite number of colliding pairs $M_1, M_2 \in \{0,1\}^*$ such that $h(M_1)=h(M_2)$.
- **The second conflict:**
  - although there are infinite numbers of colliding pairs, practical requirements to the designer of the cryptographic hash function are that it is ***infeasible in practice*** to find at least one colliding pair.

| Theoretical facts | Practical requirements |
|---|---|
| No one knows if one-way functions exist, because if they exist then we will know that P $\neq$ NP. Moreover, since we do not know if P $\neq$ NP implies that one-way functions exist, proving the existence of one-way functions is even harder than proving that P $\neq$ NP. | Construct a one-way function! |
| For any function $h : \{0,1\}^* \rightarrow \{0,1\}^n$, there exist infinite number of colliding pairs $M_1, M_2 \in \{0,1\}^*$ such that $h(M_1) = h(M_2)$. | Although there are infinite number of colliding pairs, practical finding of at least one such a pair should be infeasible! |

**TABLE 1.1**

Theoretical facts or knowledge versus practical requirements for cryptographic hash functions.

# Conflicts theory vs. practice

- The conflicts between our theoretically limited knowledge for one-way functions and the practical requirements, **make the construction of cryptographic hash functions extremely hard.**
- The task for designers is additionally hardened by additional security requirements:
  - pseudo-randomness when it is keyed with a secret key
  - near-collision resistance
  - partial preimage resistance
  - indifferentiable from random oracle

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- **Random oracle paradigm**
- **Formal definition of cryptographic hash functions and basic attacks**
- **Choosing the length of hash functions**
- **Merkle-Damgård iterative construction of hash functions**
- **A short history of practical hash functions constructions (and their failures)**
- **Merkle-Damgård Construction does not act as Random Oracle**
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

# Another point of view to cryptographic hash functions

- The **"Random-Oracle paradigm"**, Bellare and Rogaway, 1993

## Hash functions are random functions

- Digest d=h(M) are chosen uniformly for each message M
- Digest d=h(M) has no correlation with M
- For distinct $M_1, M_2, \ldots$, digests $d_i = h(M_i)$ are completely uncorrelated to each other
- Cannot find collisions, or even near-collisions
- Cannot find M to "hit" a specific d
- Cannot find fixed-points d = h(d)
- ...

NTNU
Innovation and Creativity

# Random oracle paradigm

- The **"Random-Oracle paradigm"** works like this:
  1. Assume that in your system (protocol, algorithm, ...) the hash function that you are using is really that good (as a random function)
  2. Try to prove the security of your system with that assumption.
  3. Replace the ideal hash function with a particularly specified function.
- In practice – very useful paradigm (OAEP encryption, Full Domain Hash i.e. FDH-RSA, Probabilistic Signature Schemes i.e. PSS signatures, …)

**BUT**

# Random oracle paradigm

**The controversies with the definition of cryptographic hash functions follow this paradigm too!**

- Canetti, Goldreich and Halevi: “The random oracle methodology, revisited”, 1998.
- There exists a protocol (pretty artificial, but still theoretically sound) which is provably secure in the random oracle model, **but becomes insecure when the hash function used in the protocol is replaced by any concretely specified hash function.**

NTNU
Innovation and Creativity

# Random oracle paradigm

**The controversies with the definition of cryptographic hash functions follow this paradigm too!**

- Stinson 2001:
  1. A proof in the random oracle model is therefore no more than plausible evidence of security when the random oracle model is replaced by a particular hash function.
  2. This was the “thesis” proposed by Bellare and Rogaway when they introduced the random oracle model.
  3. In favor of this thesis, it should be noted that no practical protocol proven secure in the random oracle model has been broken when used with a “good” hash function.
  4. On the other hand, the Canneti-Goldreich-Halevi result indicates that there is not likely to be any proof that this thesis is always valid.

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- **Formal definition of cryptographic hash functions and basic attacks**
- **Choosing the length of hash functions**
- **Merkle-Damgård iterative construction of hash functions**
- **A short history of practical hash functions constructions (and their failures)**
- **Merkle-Damgård Construction does not act as Random Oracle**
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

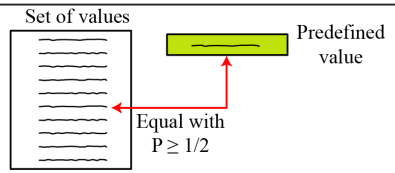
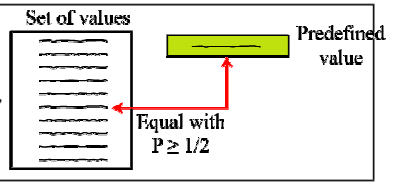
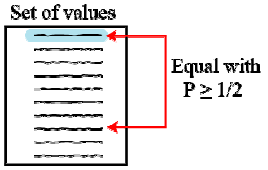
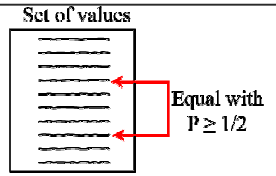
# Formal definition of hash functions and basic attacks

- Formalization used by Stinson in: “Some observations on the theory of cryptographic hash functions”, 2001 (based on Bellare-Rogaway random oracle model)
- An $(N, M)$ hash function is any function $f$: $X \rightarrow Y$, where $X$ and $Y$ are finite sets with $|X|=N$, and $|Y|=M$. These hash functions with finite domains are also called compression functions.

# Formal definition of hash functions and basic attacks (cont.)

## Four Problems

We define four problems which will be studied in the rest of the paper. In reference to Problem 1, we assume that $0 \in Y$.

| | | |
|---|---|---|
| Problem 1: | Zero preimage |  |
| Instance: | A hash function $f : X \to Y$. | |
| Find: | $x \in X$ such that $f(x) = 0$. | |
| Problem 2: | Preimage |  |
| Instance: | A hash function $f : X \to Y$ and an element $y \in Y$. | |
| Find: | $x \in X$ such that $f(x) = y$. | |
| Problem 3: | Second preimage |  |
| Instance: | A hash function $f : X \to Y$ and an element $x \in X$. | |
| Find: | $x' \in X$ such that $x' \neq x$ and $f(x') = f(x)$. | |
| Problem 4: | Collision |  |
| Instance: | A hash function $f : X \to Y$. | |
| Find: | $x, x' \in X$ such that $x' \neq x$ and $f(x') = f(x)$. | |









NTNU
Innovation and Creativity

# Formal definition of hash functions and basic attacks (cont.)

**Algorithm 2.1:** *FindZeroPreimage*$(f, q)$

choose $X_0 \subseteq X, |X_0| = q$
**for each** $x \in X_0$
  **do** { **if** $f(x) = 0$
    **then return** $(x)$
**return** (failure)

**Theorem 2.2** *For any* $X_0 \subseteq X$ *with* $|X_0| = q$, *the success probability of Algorithm 2.1 is* $\epsilon = 1 - (1 - 1/M)^q$.

For an $n$-bit hash function (i.e. $M=2^n$)
in order $\epsilon \geq 0.5$, $q \approx 0.69 \cdot 2^n$

# Formal definition of hash functions and basic attacks (cont.)

**Algorithm 2.2:** *FindPreimage*$(f, y, q)$

choose $X_0 \subseteq X, |X_0| = q$
**for each** $x \in X_0$
  **do** { **if** $f(x) = y$
    **then return** $(x)$
**return** (failure)

The success probability of Algorithm 2.2, for any fixed $y$, is the same as that of Algorithm 2.1. Therefore, the success probability averaged over all $y \in Y$ is identical, too.

**Theorem 2.3** *For any $X_0 \subseteq X$ with $|X_0| = q$, the success probability of Algorithm 2.2 is $\epsilon = 1 - (1 - 1/M)^q$.*

For an $n$-bit hash function (i.e. $M=2^n$)
in order $\square \geq 0.5$, $q \approx 0.69 \cdot 2^n$

# Formal definition of hash functions and basic attacks (cont.)

**Algorithm 2.3:** *FindSecondPreimage*$(f, x, q)$

$y \leftarrow f(x)$
choose $X_0 \subseteq X \backslash \{x\}, |X_0| = q - 1$
**for each** $x_0 \in X_0$
  **do** { **if** $f(x_0) = y$
    **then return** $(x_0)$
**return** (failure)

The analysis of Algorithm 2.3 is similar to the two previous algorithms. The only difference is that we require an "extra" application of $f$ to compute $y = f(x)$ for the input value $x$.

**Theorem 2.4** *For any* $X_0 \subseteq X \backslash \{x\}$ *with* $|X_0| = q - 1$, *the success probability of Algorithm 2.3 is* $\epsilon = 1 - (1 - 1/M)^{q-1}$.

For an $n$-bit hash function (i.e. $M=2^n$)
in order $\epsilon \geq 0.5$, $q \approx 0.69 \cdot 2^n + 1$

# Formal definition of hash functions and basic attacks (cont.)

**Algorithm 2.4:** *FindCollision*$(f, q)$

choose $X_0 \subseteq X \backslash \{x\}, |X_0| = q$
**for each** $x \in X_0$
  **do** $y_x \leftarrow f(x)$
**if** $y_x = y_{x'}$ for some $x' \neq x$
  **then return** $(x, x')$
  **else return** (failure)

**Theorem 2.5** *For any $X_0 \subseteq X$ with $|X_0| = q$, the success probability of Algorithm 2.4 is*

$$\epsilon = 1 - \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-q+1}{M}\right).$$

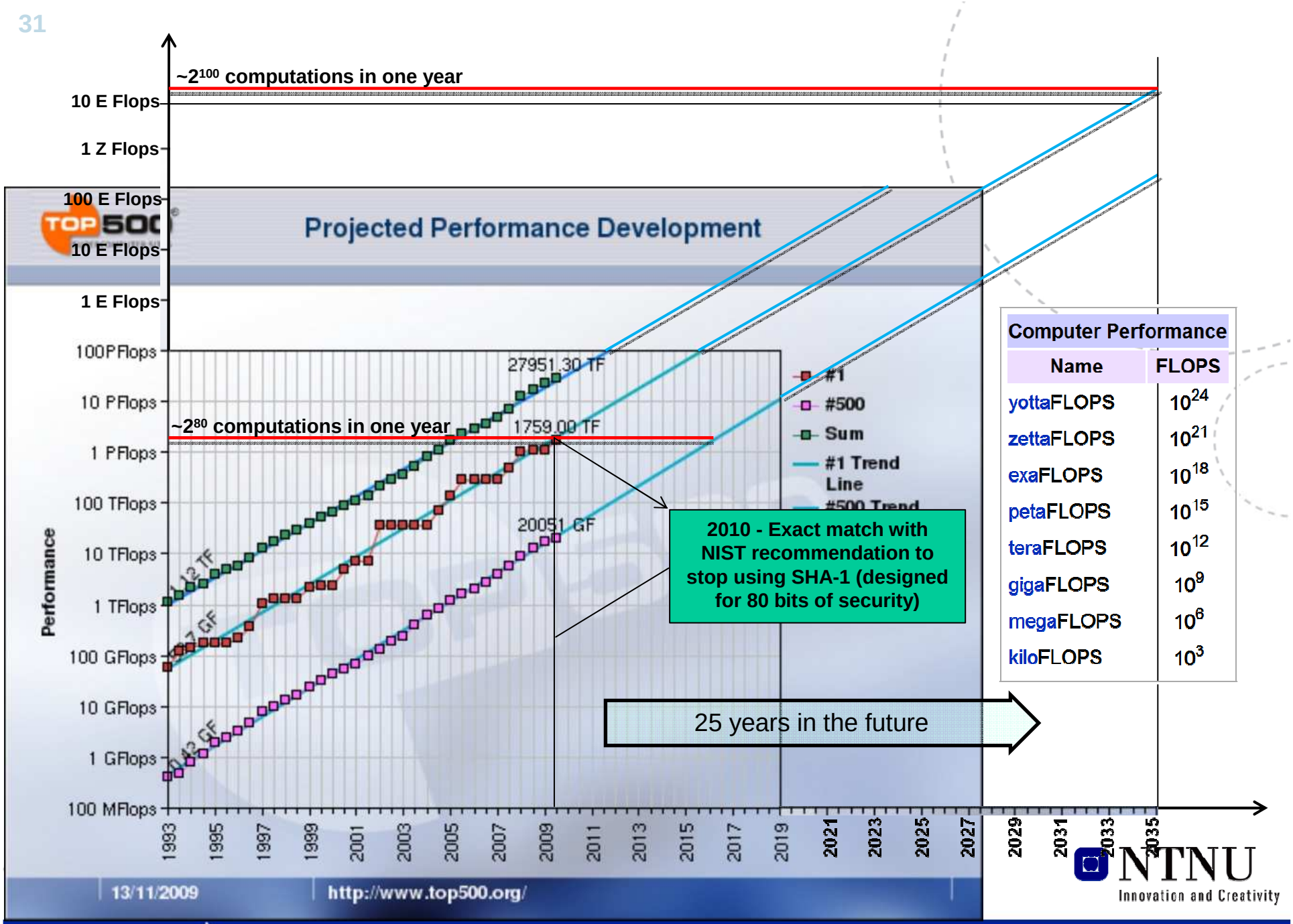
For an $n$-bit hash function (i.e. $M=2^n$)
in order $\epsilon \geq 0.5$, $q \approx 1.18 \cdot 2^{n/2}$

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- **Choosing the length of hash functions**
- **Merkle-Damgård iterative construction of hash functions**
- **A short history of practical hash functions constructions (and their failures)**
- **Merkle-Damgård Construction does not act as Random Oracle**
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

# Choosing the length of Hash outputs

- Because of the birthday attack, the length of hash outputs in general should double the key length of block ciphers
  - From 2010 recommended hash functions and hash output lengths are: SHA-224, SHA-256, SHA-384, SHA-512 with projected collision security of 112, 128, 192 and 256 bits.




| Computer Performance | |
|---|---|
| Name | FLOPS |
| yottaFLOPS | $10^{24}$ |
| zettaFLOPS | $10^{21}$ |
| exaFLOPS | $10^{18}$ |
| petaFLOPS | $10^{15}$ |
| teraFLOPS | $10^{12}$ |
| gigaFLOPS | $10^{9}$ |
| megaFLOPS | $10^{6}$ |
| kiloFLOPS | $10^{3}$ |

NTNU
Innovation and Creativity

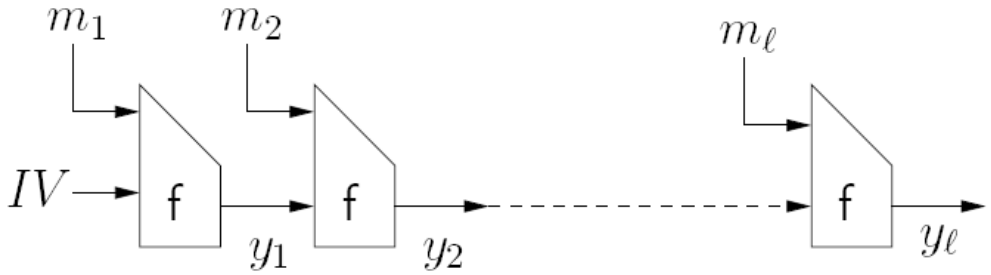
# Outline

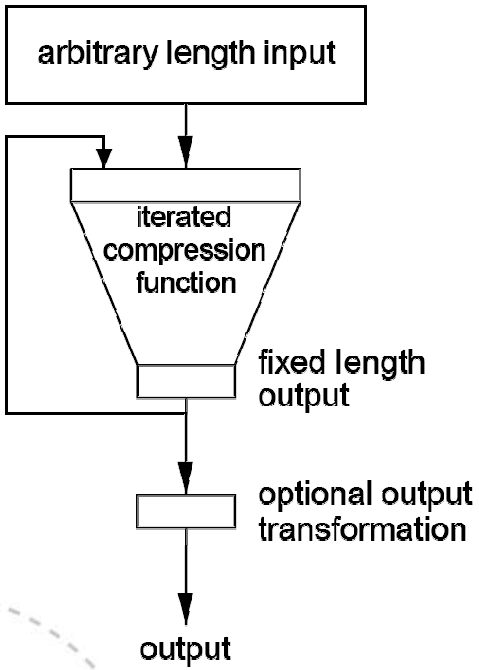
- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- **Merkle-Damgård iterative construction of hash functions**
- **A short history of practical hash functions constructions (and their failures)**
- **Merkle-Damgård Construction does not act as Random Oracle**
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

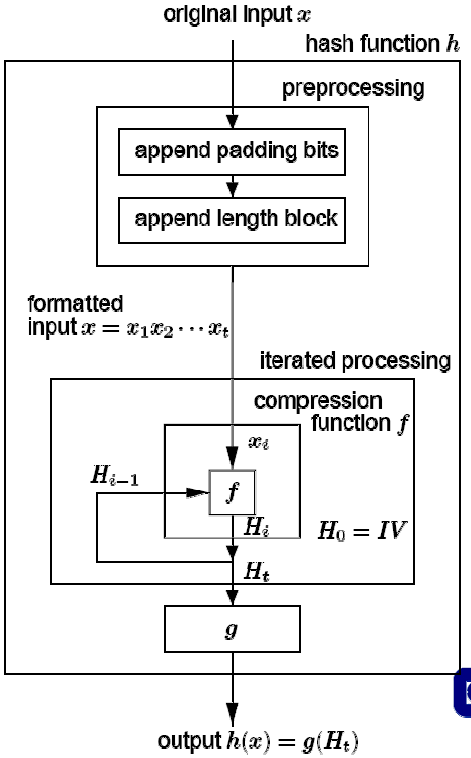
# Iterative Construction of Hash Functions (Merkle-Damgård Construction)

- A hash function needs to map a message of an arbitrary length to a $n$-bit output
  - $h$: $\{0,1\}^* \rightarrow \{0,1\}^n$
- The iterative construction
  - use a compression function that takes a fixed-length input string and output a shorter string
    - $f$:$\{0,1\}^{n+t} \rightarrow \{0,1\}^n$
  - a message is divided into fixed length blocks and processed block by block
  - Merkle and Damgård in 1989 (independently) showed that given a collision resistant compression function, a collision resistant hash function could be constructed

# Iterative Construction of Hash Functions (Merkle-Damgård Construction)



# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- **A short history of practical hash functions constructions (and their failures)**
- **Merkle-Damgård Construction does not act as Random Oracle**
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

# A short history of practical hash functions constructions (and their failures)

- **MD**x family
  - Proposed by **Ronald Rivest** for **RSA** Labs
  - **MD2:** 1989, Broken
  - **MD4**, 1990, Broken
  - **MD5**, 1992, Broken

- **Inspired by MD**x family
  - RIPEMD: 1995, Broken
  - RIPEMD-160: 1996
  - Haval: 1992, Broken

**SHA** family, (**SHA**: **S**ecure **H**ash **A**lgorithm) – also inspired by MDx family

− Developed by **NSA** (**N**ational **S**ecurity **A**gency)

**SHA-0**, 1993, FIPS-180, US Gov., Broken
**SHA-1**, 1995, FIPS-180-1, US Gov., Theoretically broken in 2005
**SHA-2,** 2002, FIPS-180-2, US Gov., SHA-224/256/384/512

NTNU
Innovation and Creativity

# A short history of practical hash functions constructions (and their failures)

- **Serious flaws in MD4 and MD5** [RIPE '91-'92]
- **SHA replaced by SHA-1** [NSA '94]
- **Collisions for MD4, problem in ext.-MD4** [Dobbertin '96]
- **More problems of MD5 and RIPEMD** [Dobbertin '96]

- **Collisions for Haval** [Biryukov, Van Rompay, Preneel '02]
- **Collisions for SHA-0** [Joux '04]
- **Collisions for MD4 (by hand), MD5, and RIPEMD** [Wang, Feng, Lai, Yu '04]
- **Attack on 53 out of 80 rounds of SHA-1** [Oswald-Rijmen'04 and Biham-Chen] '04]
- **$2^{39}$ attack on SHA-0** [Wang,Yu,Yin '05]
- **$2^{69}$ attack on SHA-1** [Wang, Yin, Yu '05]
- **$2^{63}$ attack on SHA-1** [Wang, Yao, Yao '05]

NTNU
Innovation and Creativity

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- **Merkle-Damgård Construction does not act as Random Oracle**
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

# Merkle-Damgård Construction does not act as Random Oracle

- Length extension attack on Merkle-Damgård
- For a message $M = M_1\ M_2\ \ldots\ M_N$, and the compression function f(A, B) used in Merkle-Damgård construction, the hash function is defined as:

$$h(M)=C(C(\ldots C(IV,M_1),M_2),\ldots M_N),P_M)$$

- where $P_M$ is the proper padding.
- The attacker does not M, but can guess its length i.e. he knows the block $P_M$.
- He can easily construct a message $M'=P_M||M'_1$, such that he knows the hash of the message $M||M'$ i.e. he knows how to compute

$$H(M||M') = C(C(H(M),M_1),P_{M'})$$

# Merkle-Damgård Construction does not act as Random Oracle

- Multi-collisions (Joux 2004)

1. Set i=0
2. Set IV=$h_i$
3. After $2^{n/2}$ computations of the compression function f() find a colliding pair of different messages $(m_i^1, m_i^2)$ such that

$$h_{i+1}= f(h_i, m_i^1)=f(h_i, m_i^2)$$

4. i=i+1
5. Repeat steps 3 and 4 while i<r

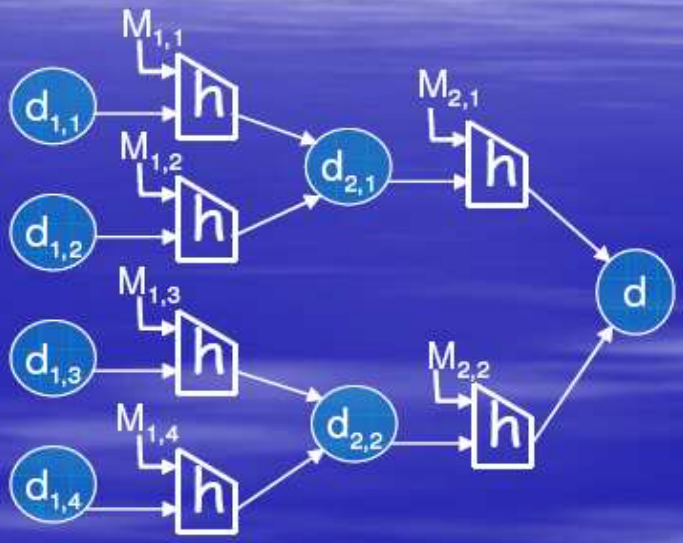
- From the pairs $(m_0^1, m_0^2)$ x $(m_1^1, m_1^2)$ x…x $(m_{r-1}^1, m_{r-1}^2)$ construct $2^r$ messages all giving $2^r$ collisions.
- Thus the complexity of finding $2^r$ collisions is just $O(r2^r)$
- If the function f is random oracle, finding $2^r$ collisions has complexity $\Theta(2^{nr(r-1)/2})$

NTNU
Innovation and Creativity

# Example attack: herding

[Kelsey-Kohno 2006]

- Find many off-line collisions
  - "Tree structure" with ~$2^{n/3}$ $d_{i,j}$'s
  - Takes ~ $2^{2n/3}$ time
- Publish final d
- Then for any prefix P
  - Find "linking block" L s.t. H(P|L) in the tree
  - Takes ~ $2^{2n/3}$ time
  - Read off the tree the suffix S to get to d




→ Show an extension of P s.t. H(P|L|S) = d

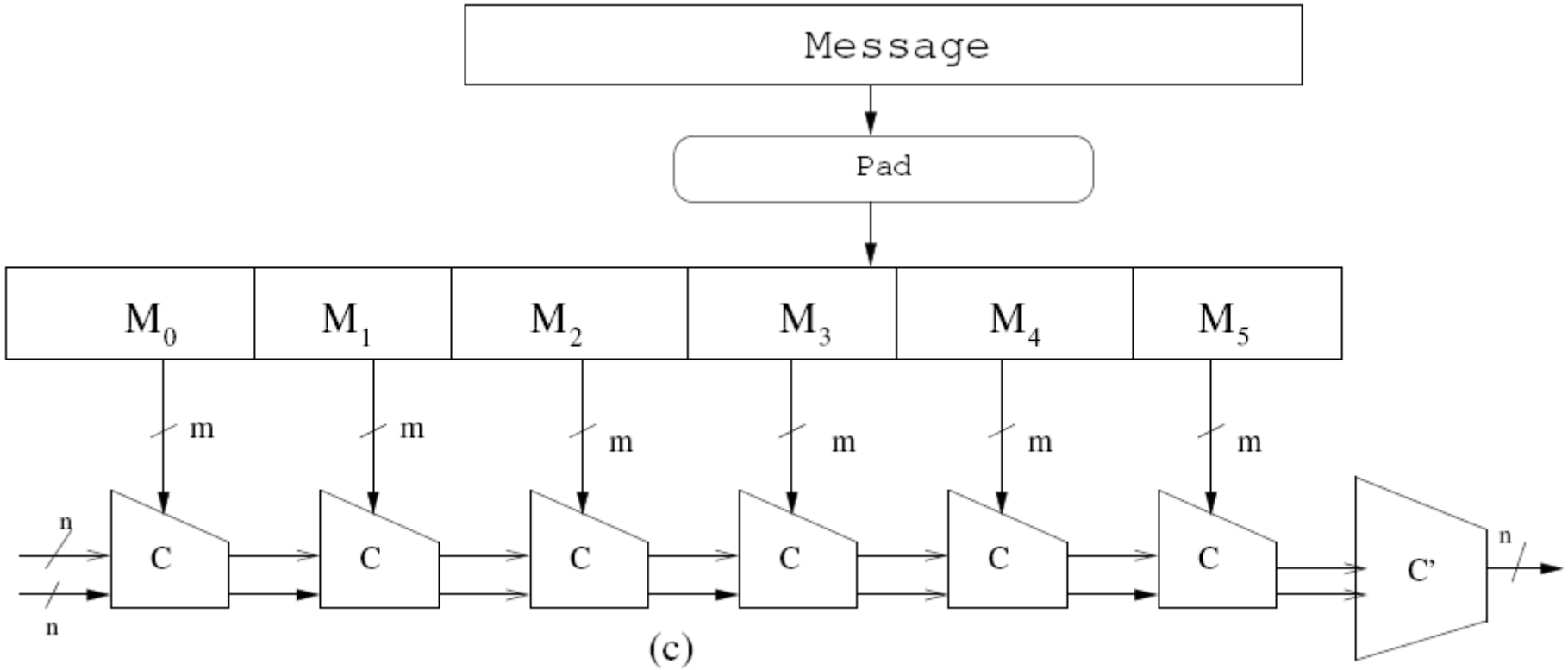
Slide, from Shai Halevi

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- Merkle-Damgård Construction does not act as Random Oracle
- **Constructions of iterative hash functions that do not suffer from attacks on MD**
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

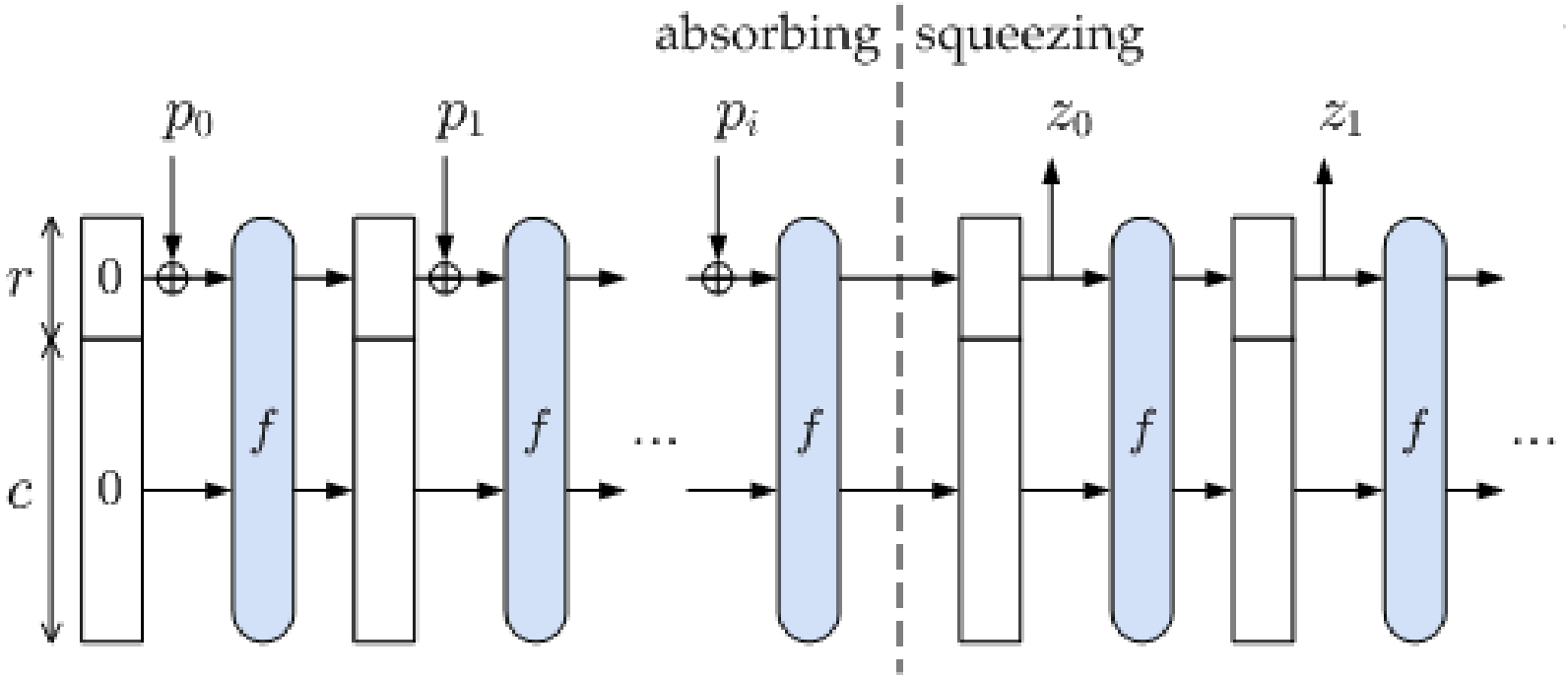
# Constructions of iterative hash functions that do not suffer from attacks on MD

- Wide pipe (Lucks 2005)



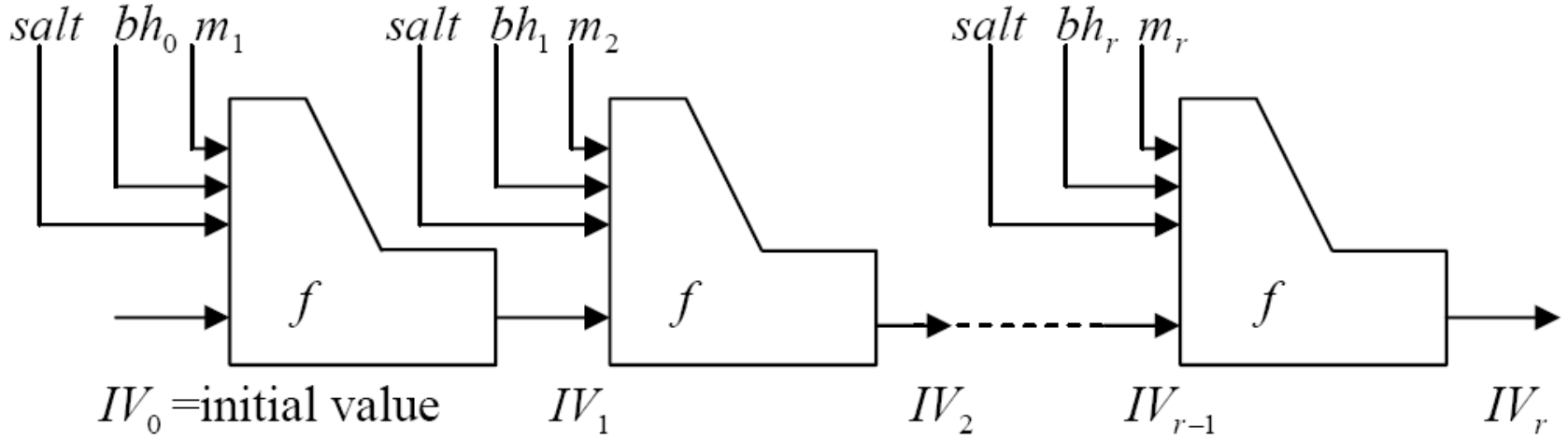
# Constructions of iterative hash functions that do not suffer from attacks on MD

- Sponge design (Bertoni, Daemen, Peeters and Van Assche, 2007)



# Constructions of iterative hash functions that do not suffer from attacks on MD

- HAIFA (Biham, Dunkelman, 2006)



# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- Merkle-Damgård Construction does not act as Random Oracle
- Constructions of iterative hash functions that do not suffer from attacks on MD
- **Building blocks in the hash functions**
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

# Building blocks in the hash functions

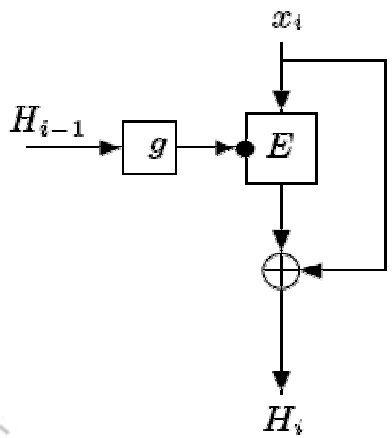
- Based on block ciphers

**(i) Single-length MDCs of rate 1**

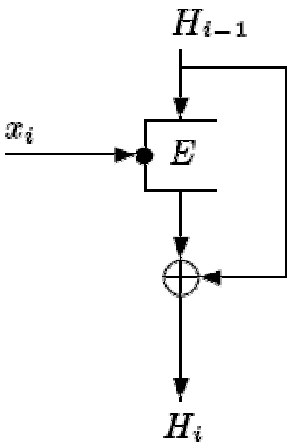
The first three schemes described below, and illustrated in Figure 9.3, are closely related single-length hash functions based on block ciphers. These make use of the following pre-defined components:

1. a generic $n$-bit block cipher $E_K$ parametrized by a symmetric key $K$;
2. a function $g$ which maps $n$-bit inputs to keys $K$ suitable for $E$ (if keys for $E$ are also of length $n$, $g$ might be the identity function); and
3. a fixed (usually $n$-bit) initial value $IV$, suitable for use with $E$.

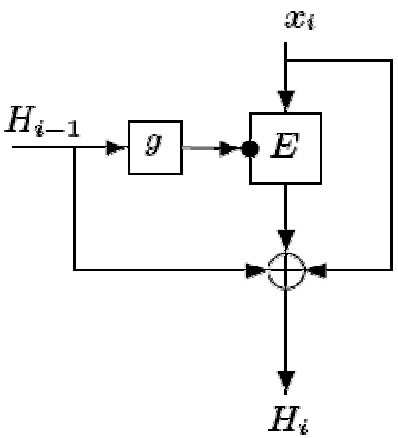
Matyas-Meyer-Oseas




Davies-Meyer

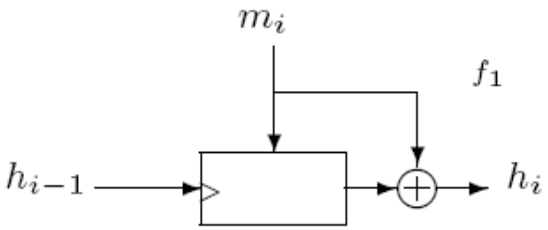


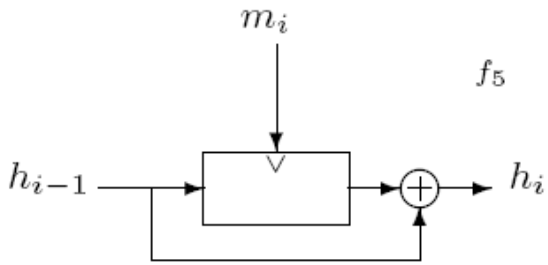

Miyaguchi-Preneel

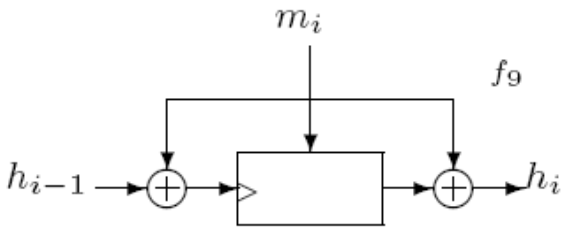


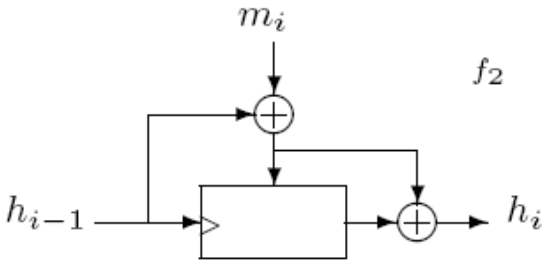
# Building blocks in the hash functions

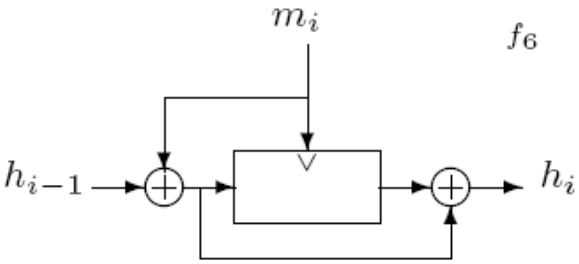
- Based on block ciphers
  - Govaerts, Preneel and Vandewalle: “Hash functions based on block ciphers: A synthetic approach”, 1993
  - 12 secure schemes (PGV1-12)
  - Strict black-box analysis done by Rogaway, Black and Shrimpton I 2002.

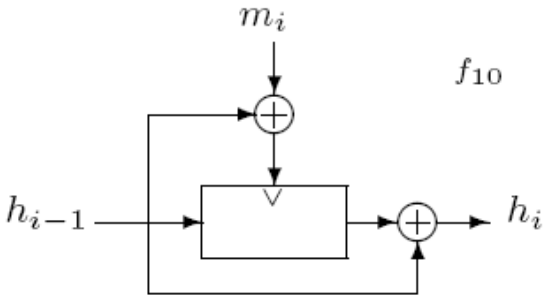

$m_i$
$f_1$
$h_{i-1}$
$h_i$

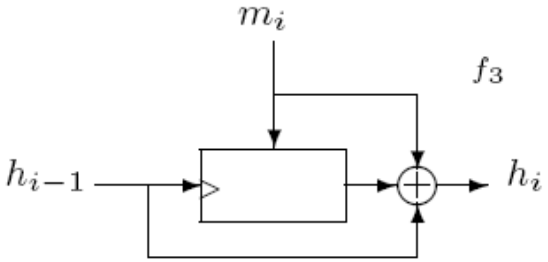

$m_i$
$f_5$
$h_{i-1}$
$h_i$

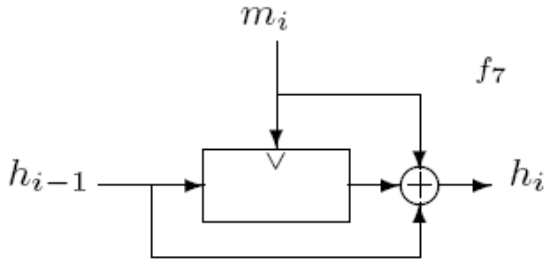

$m_i$
$f_9$
$h_{i-1}$
$h_i$

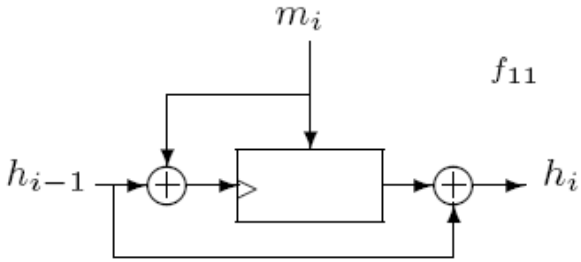

$m_i$
$f_2$
$h_{i-1}$
$h_i$

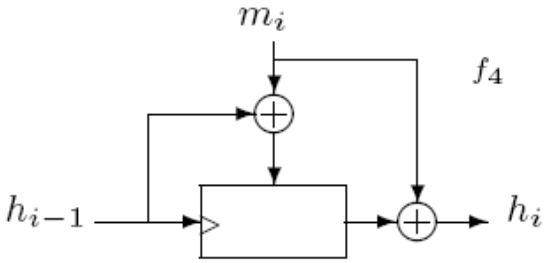

$m_i$
$f_6$
$h_{i-1}$
$h_i$

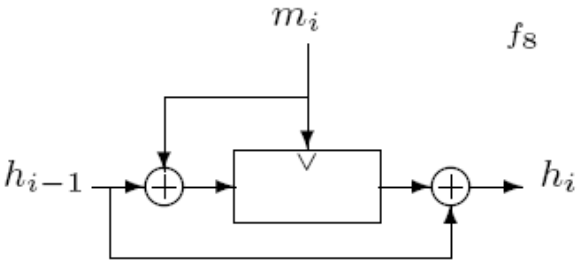

$m_i$
$f_{10}$
$h_{i-1}$
$h_i$

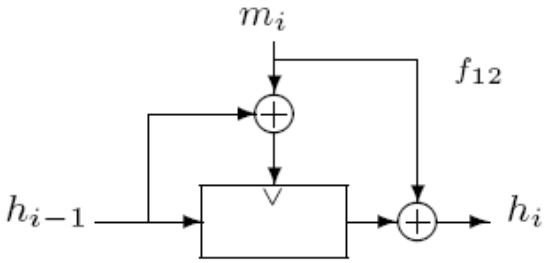

$m_i$
$f_3$
$h_{i-1}$
$h_i$


$m_i$
$f_7$
$h_{i-1}$
$h_i$


$m_i$
$f_{11}$
$h_{i-1}$
$h_i$


$m_i$
$f_4$
$h_{i-1}$
$h_i$


$m_i$
$f_8$
$h_{i-1}$
$h_i$


$m_i$
$f_{12}$
$h_{i-1}$
$h_i$

# Building blocks in the hash functions

- Based on block ciphers
  - MDC-2 and MDC-4 (based on DES, 2 or 4 invocations of the block cipher), IBM 1987 (patented, but so slow that no one used them, so IBM did not renew their patent for USA and Europe and the patent expired in 2002).

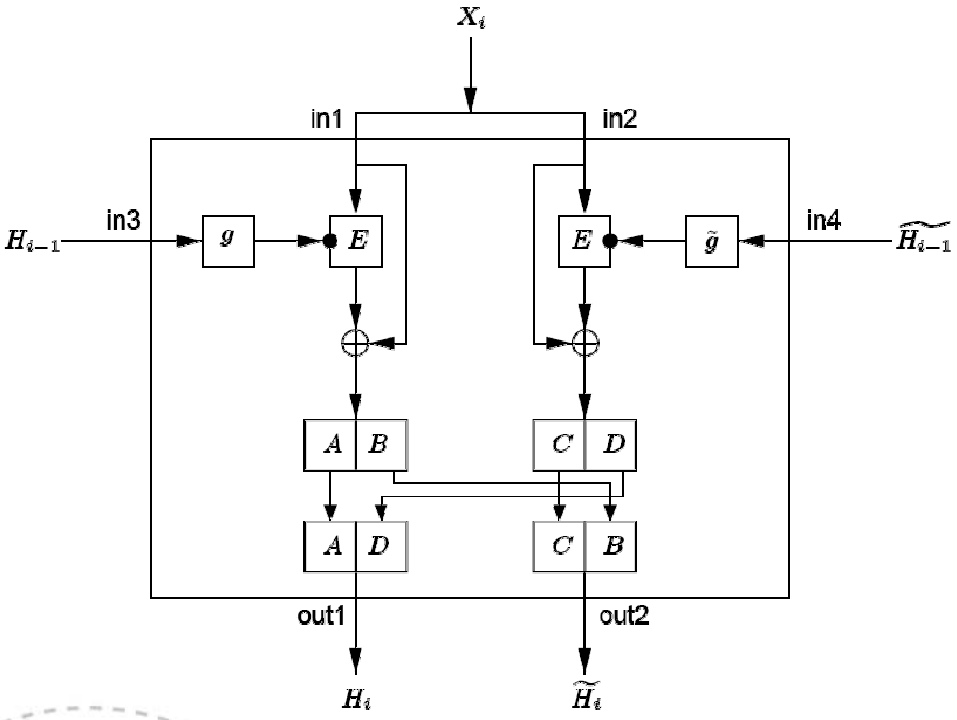



**Figure 9.4:** *Compression function of MDC-2 hash function. $E$ = DES.*

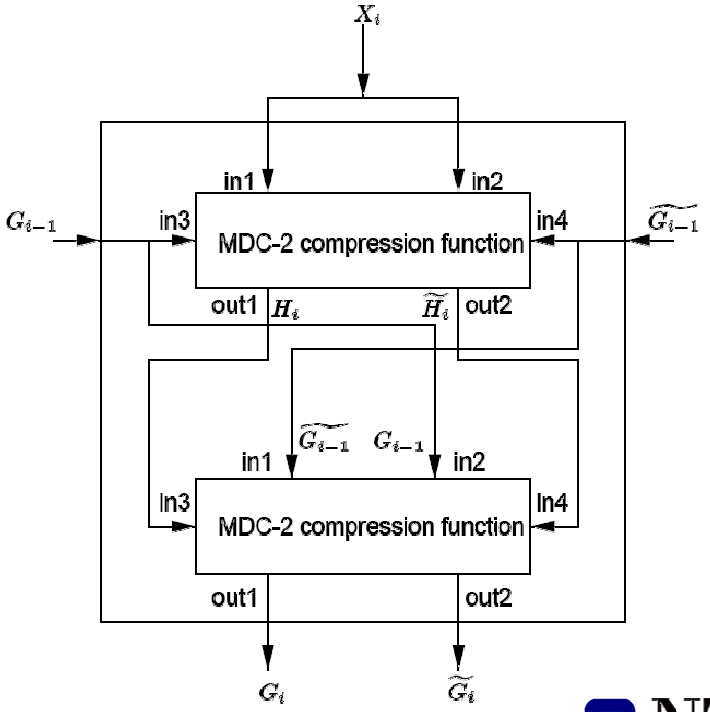



**Figure 9.5:** *Compression function of MDC-4 hash function*

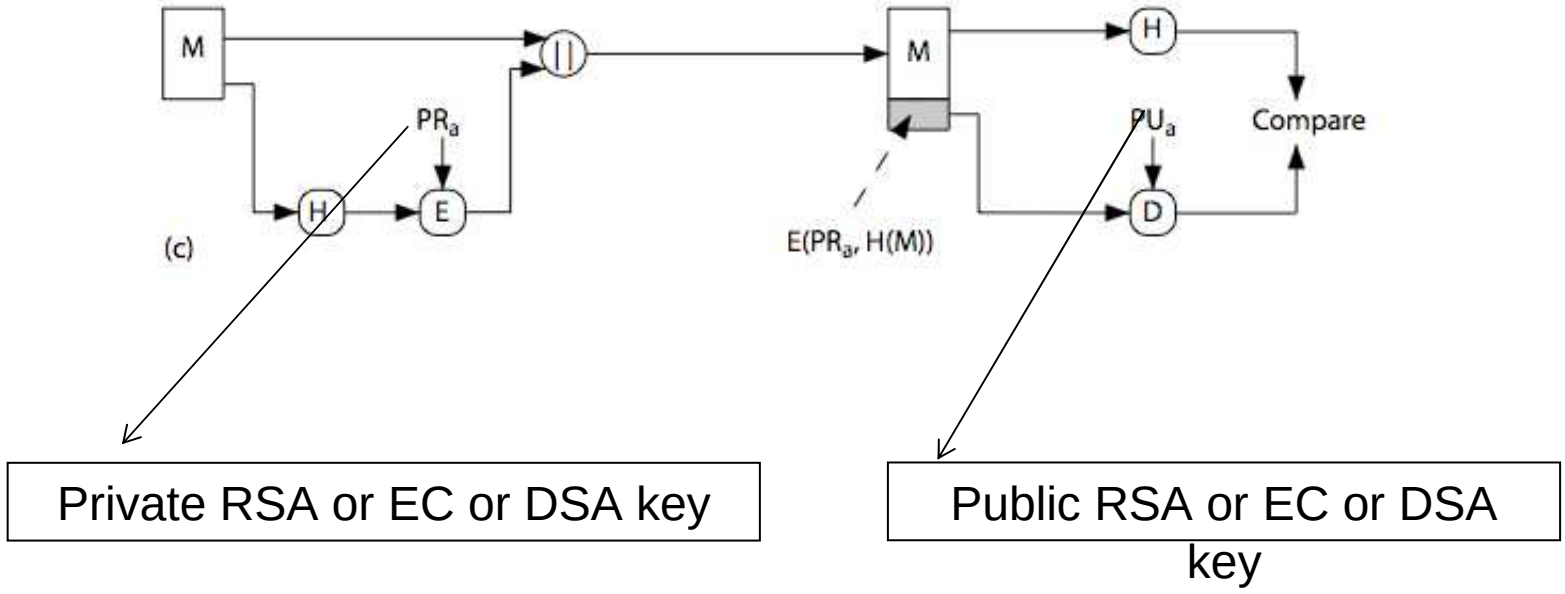
NTNU
Innovation and Creativity

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- Merkle-Damgård Construction does not act as Random Oracle
- Constructions of iterative hash functions that do not suffer from attacks on MD
- Building blocks in the hash functions
- **Description of the main applications of hash functions**
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

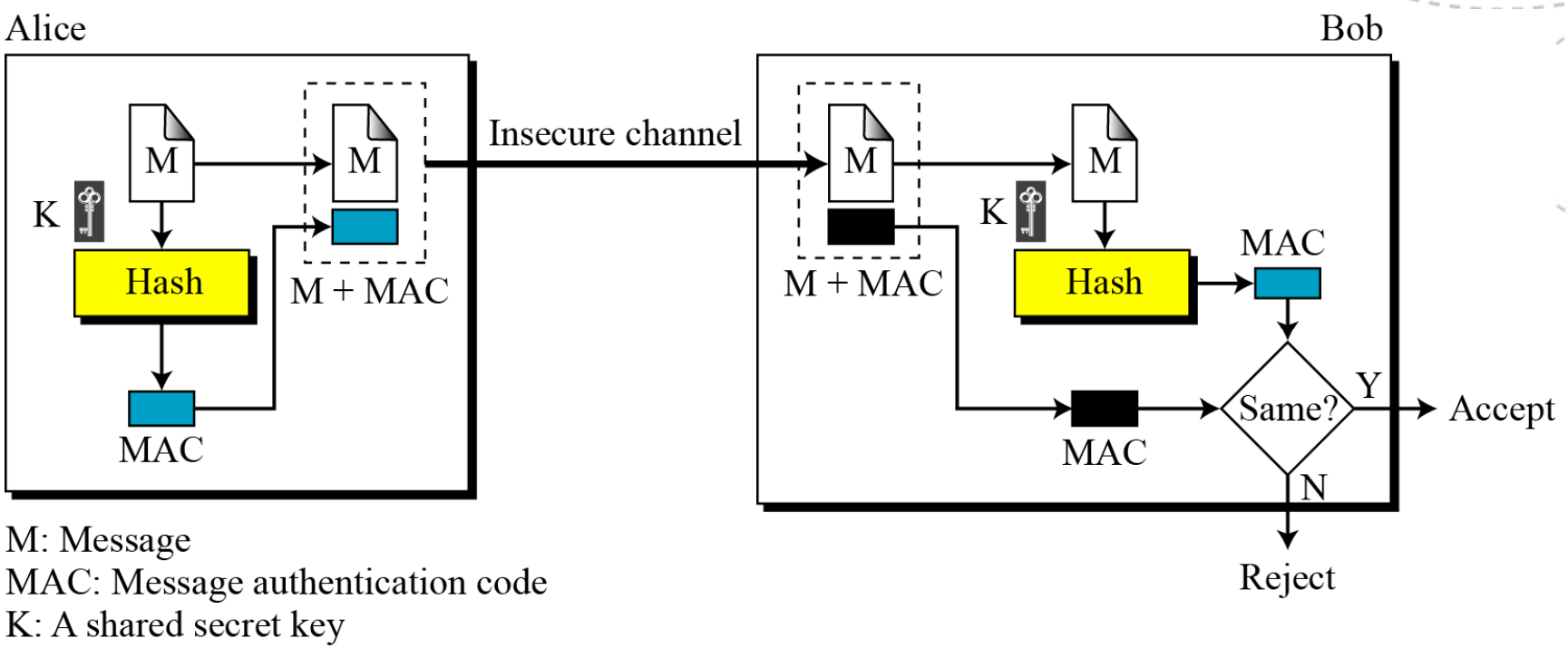
# Description of the main applications of hash functions

- Digital signatures



# Description of the main applications of hash functions

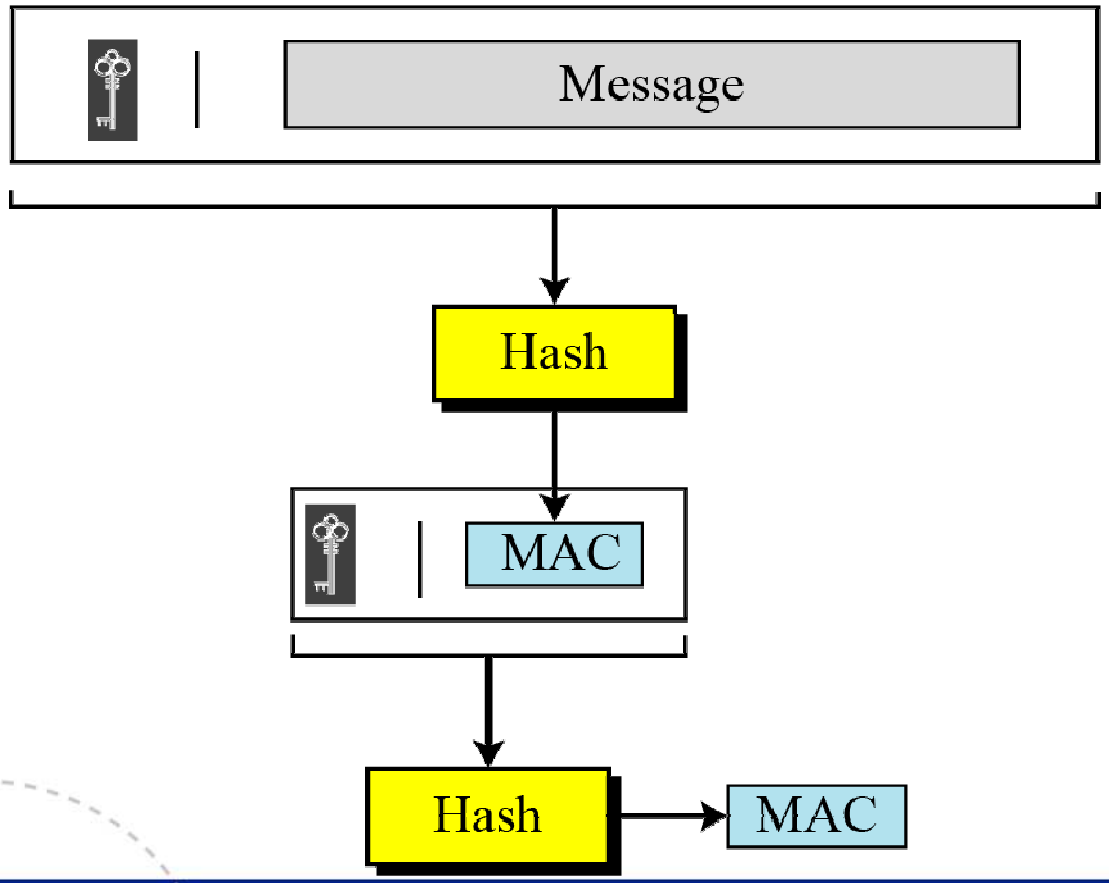
- MAC and HMAC




M: Message
MAC: Message authentication code
K: A shared secret key

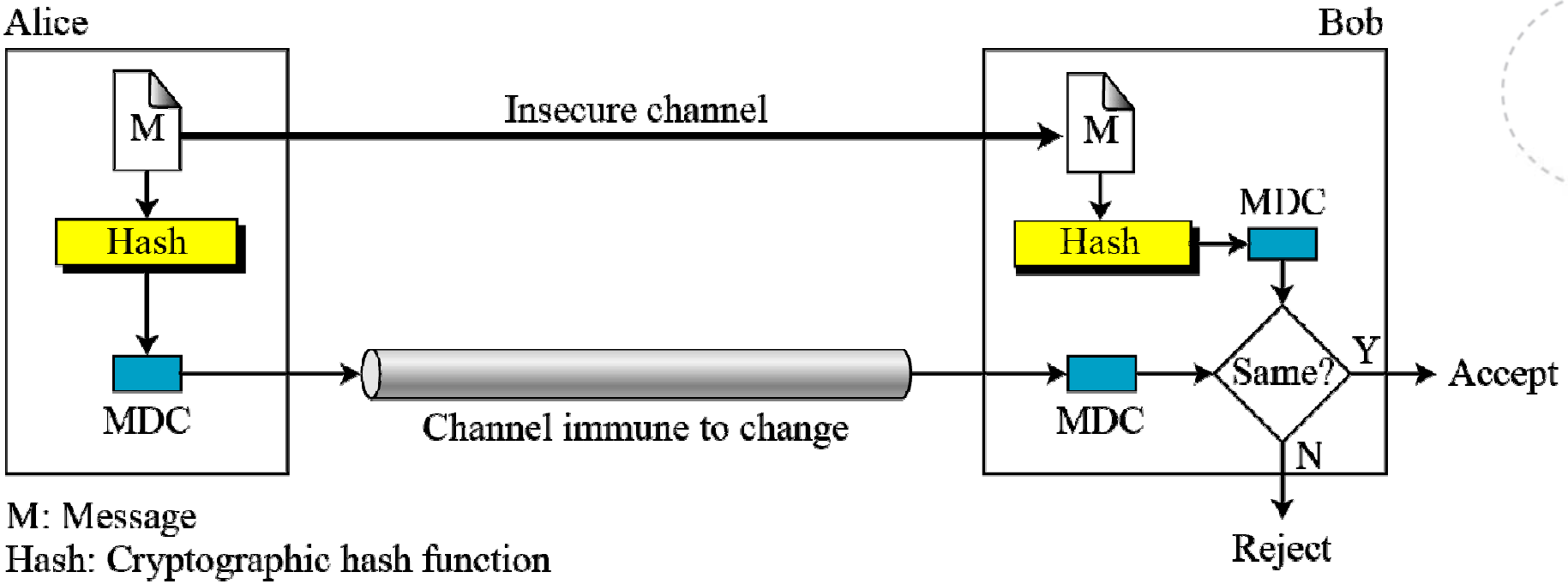
# Description of the main applications of hash functions

- MAC and HMAC



# Description of the main applications of hash functions

- Modification Detection Codes (MDC)



# Description of the main applications of hash functions

**NIST Special Publication 800-108**

**Recommendation for Key Derivation Using Pseudorandom Functions**

**(Revised)**

**Lily Chen**

**Computer Security Division**
**Information Technology Laboratory**

**C O M P U T E R  S E C U R I T Y**

**October 2009**

**Uses hash functions, via HMAC**

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- Merkle-Damgård Construction does not act as Random Oracle
- Constructions of iterative hash functions that do not suffer from attacks on MD
- Building blocks in the hash functions
- Description of the main applications of hash functions
- **NIST worldwide hash competition for SHA-3**
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

# NIST worldwide hash competition for SHA-3


NTNU
Innovation and Creativity

**Preliminary Determination by the ITC**

The ITC will preliminarily determine, within 25 days after the date on which it receives notice of the initiation, whether there is a reasonable indication that imports of subsidized LWTP from the PRC are causing material injury, or threatening to cause material injury, to a U.S. industry. *See* section 703(a)(2) of the Act. A negative ITC determination will result in the investigation being terminated; otherwise, the investigation will proceed according to statutory and regulatory time limits.

This notice is issued and published pursuant to section 777(i) of the Act.

Dated: October 29, 2007.

Stephen J. Claeys,

*Acting Assistant Secretary for Import Administration.*

[FR Doc. E7–21616 Filed 11–1–07; 8:45 am]

BILLING CODE 3510–DS–S

## DEPARTMENT OF COMMERCE

### National Institute of Standards and Technology

**[Docket No.: 070911510–7512–01]**

### Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA–3) Family

**AGENCY:** National Institute of Standards and Technology, Commerce.

**ACTION:** Notice and request for nominations for candidate hash algorithms.

**SUMMARY:** This notice solicits nominations from any interested party for candidate algorithms to be considered for SHA–3, and specifies how to submit a nomination package. It presents the nomination requirements and the minimum acceptability requirements of a "complete and proper" candidate algorithm submission. The evaluation criteria that will be used to appraise the candidate algorithms are also described.

**DATES:** Candidate algorithm nomination packages must be received by October 31, 2008. Further details are available in section 2.

**ADDRESSES:** Candidate algorithm submission packages should be sent to: Ms. Shu-jen Chang, Information Technology Laboratory, *Attention:* Hash Algorithm Submissions, 100 Bureau Drive—Stop 8930, National Institute of Standards and Technology, Gaithersburg, MD 20899–8930.

**FOR FURTHER INFORMATION CONTACT:** For general information, send e-mail to *hash-function@nist.gov*. For questions related to a specific submission package, contact Ms. Shu-jen Chang, National Institute of Standards and Technology, 100 Bureau Drive—Stop 8930, Gaithersburg, MD 20899–8930; telephone: 301–975–2940 or via fax at 301–975–8670, e-mail: *shu-jen.chang@nist.gov*.

**SUPPLEMENTARY INFORMATION:** This notice contains the following sections:



NIST expects SHA–3 to have a security strength that is at least as good as the hash algorithms currently specified in FIPS 180–2, and that this security strength will be achieved with significantly improved efficiency. NIST

digest could act as a proxy for a possibly very large variable length message in a digital signature algorithm, such as RSA or DSA. These hash functions have since been widely used for many other "ancillary" applications, including hash-based message authentication codes, pseudo random number generators, and key derivation functions.

A series of related hash functions have been developed, such as MD4, MD5, SHA–0, SHA–1 and the SHA–2 family, (which includes 224, 256, 384 and 512-bit variants); all of these follow the Merkle-Damgard construct. NIST began the standardization of the SHA hash functions in 1993, with a specification of SHA–0 in the Federal Information Processing Standards Publication (FIPS PUBS) 180, the Secure Hash Standard; subsequent revisions of the FIPS have replaced SHA–0 with SHA–1 and added the SHA–2 family in FIPS 180–1 and FIPS 180–2, respectively.

Recently, cryptanalysts have found collisions on the MD4, MD5, and SHA–0 algorithms; moreover, a method for finding SHA–1 collisions with less than the expected amount of work has been published, although at this time SHA–1 collisions have not yet been demonstrated. Although there is no specific reason to believe that a practical attack on any of the SHA–2 family of hash functions is imminent, a successful collision attack on an algorithm in the SHA–2 family could have catastrophic effects for digital signatures.

NIST has decided that it is prudent to develop a new hash algorithm to augment and revise FIPS 180–2. The new hash algorithm will be referred to

Therefore, the submitted algorithms for SHA–3 must provide message digests of 224, 256, 384 and 512 bits to allow substitution for the SHA–2 family. The 160-bit hash value produced by SHA–1 is becoming too small to use for digital signatures, therefore, a 160-bit replacement hash algorithm is not contemplated.

Many cryptographic applications that are currently specified in FIPS and NIST Special Publications require the use of a NIST-approved hash algorithm. These publications include:

- FIPS 186–2, Digital Signature Standard;
- FIPS 198, The Keyed-Hash Message Authentication Code (HMAC);
- SP 800–56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography; and
- SP 800–90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators (DRBGs).



The SHA–3 algorithm is expected to be suitable for these applications.

Since SHA–3 is expected to provide a simple substitute for the SHA–2 family of hash functions, certain properties of the SHA–2 hash functions must be preserved, including the input parameters; the output sizes; the collision resistance, preimage resistance, and second-preimage resistance properties; and the "one-pass" streaming mode of execution. However, it is also desirable that the selected SHA–3 algorithm offer features or properties that exceed, or improve upon, the SHA–2 hash functions. For example, the selected SHA–3 algorithm may offer efficient integral options, such as randomized hashing, that

For interoperability, NIST strongly desires a single hash algorithm family (that is, that different size message digests be internally generated in as similar a manner as possible) to be selected for SHA–3. However, if more than one suitable candidate family is identified, and each provides significant advantages, NIST may consider recommending more than one family for inclusion in the revised Secure Hash Standard.

### 2. Requirements for Candidate Algorithm Submission Packages

Candidate algorithm nomination packages must be received by October 31, 2008. Submission packages received before August 31, 2008 will be reviewed for completeness by NIST; the submitters will be notified of any deficiencies by September 30, 2008, allowing time for deficient packages to be amended by the submission deadline. No amendments to packages will be permitted after the submission deadline. Requests for the withdrawal of submission packages will only be honored until the submission deadline.

Due to the specific requirements of the submission package such as Intellectual Property Statements / Agreements / Disclosures as specified in section 2.D, e-mail submissions will not be accepted for these statements or for the initial submission package. However, e-mail submissions of amendments to the initial submission package will be allowed prior to the submission deadline.

"Complete and proper" submission packages received in response to this notice will be posted at *http://www.nist.gov/hash-competition* for inspection. To be considered as a

ecifications and mentation.

roperty Statements/ osures.

hission Requirements.

s is discussed in

all contain the tion:

ubmitted algorithm.

mitter's name, e-mail e, fax, organization,

xiliary submitter(s).

- Name of the algorithm inventor(s)/developer(s).
- Name of the owner, if any, of the algorithm. (normally expected to be the same as the submitter).
- Signature of the submitter.
- (optional) Backup point of contact (with telephone, fax, postal address, e-mail address).

#### *2.B Algorithm Specifications and Supporting Documentation*

2.B.1 A complete written specification of the algorithm shall be included, consisting of all necessary mathematical operations, equations, tables, diagrams, and parameters that are needed to implement the algorithm. The document shall include design rationale (e.g., the rationale for choosing the specific number of rounds for computing the hashes) and an explanation for all the important design decisions that are made. It should also include 1) any security argument that is applicable, such as a security reduction proof, and 2) a preliminary analysis, such as possible attack scenarios for collision-finding, first-preimage-finding, second-preimage-finding, length-extension attack, multicollision attack, or any cryptographic attacks that have been considered and their results.

In addition, the submitted algorithm may include a tunable security parameter, such as the number of rounds, which would allow the selection of a range of possible security/performance tradeoffs. If such a parameter is provided, the submission document must specify a recommended value for each digest size specified in Section 3, with justification. The submission should also provide any bounds that the designer feels are appropriate for the parameter, including a bound below which the submitter expects cryptanalysis to become practical. The tunable parameter may be used to produce weakened versions of the submitted algorithm for analysis, and permit NIST to select a different security/performance tradeoff than originally specified by the submitter, in light of discovered attacks or other analysis, and in light of the alternative algorithms that are available. NIST will consult with the submitter of the algorithm if it plans to select that algorithm for SHA–3, but with a different parameter value than originally specified by the submitter. Submissions that do not include such a parameter should include a weakened version of the submitted algorithm for analysis, if at all possible.

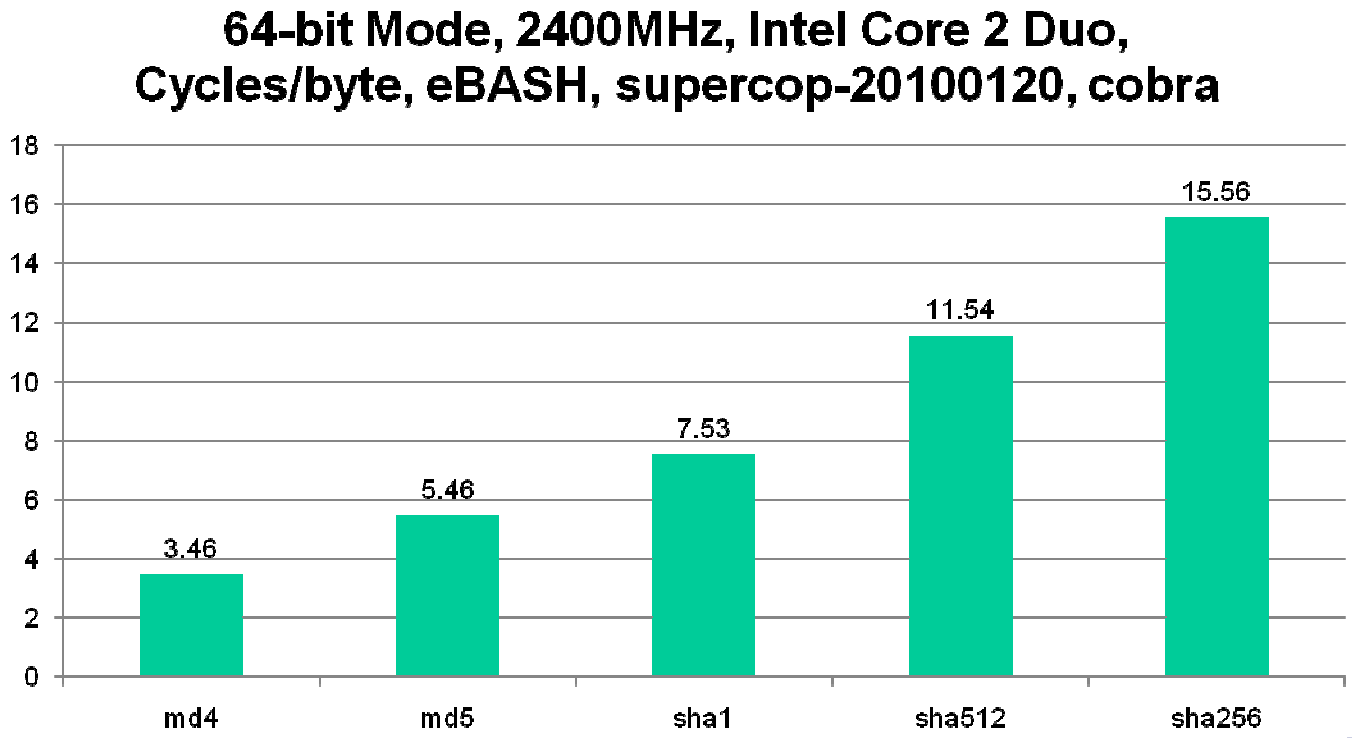
NIST is open to, and encourages, submissions of hash functions that differ from the traditional Merkle-Damgard model, using other structures, chaining modes, and possibly additional inputs. However, if a submitted algorithm cannot be used directly in current applications of hash functions as specified in FIPS or NIST Special Publications, the submitted algorithm must define a compatibility construct with the same input and output parameters as the SHA hash functions such that it can replace the existing SHA functions in current applications without any loss of security. The replacement of all SHA functions in any standardized application by this compatibility construct shall require no additional modification of the standard application beyond the alteration of any algorithm specific parameters already present in the standard, such as algorithm name and message block length. Submissions may optionally define other variants, constructs, or iterated structures for specific useful applications.

# NIST worldwide hash competition for SHA-3

- 64 Submissions
- 51 Remained in the Round 1 as proper and complete
- First Cryptographic workshop to list all the candidates was held at Katolike Universitet of Lueven Belgium (25-26 February 2008)
- In July 2009 NIST reduced the number of candidates to 14

# NIST worldwide hash competition for SHA-3

- Why NIST expects SHA-3 to be “with significantly improved efficiency” (over SHA-2)?



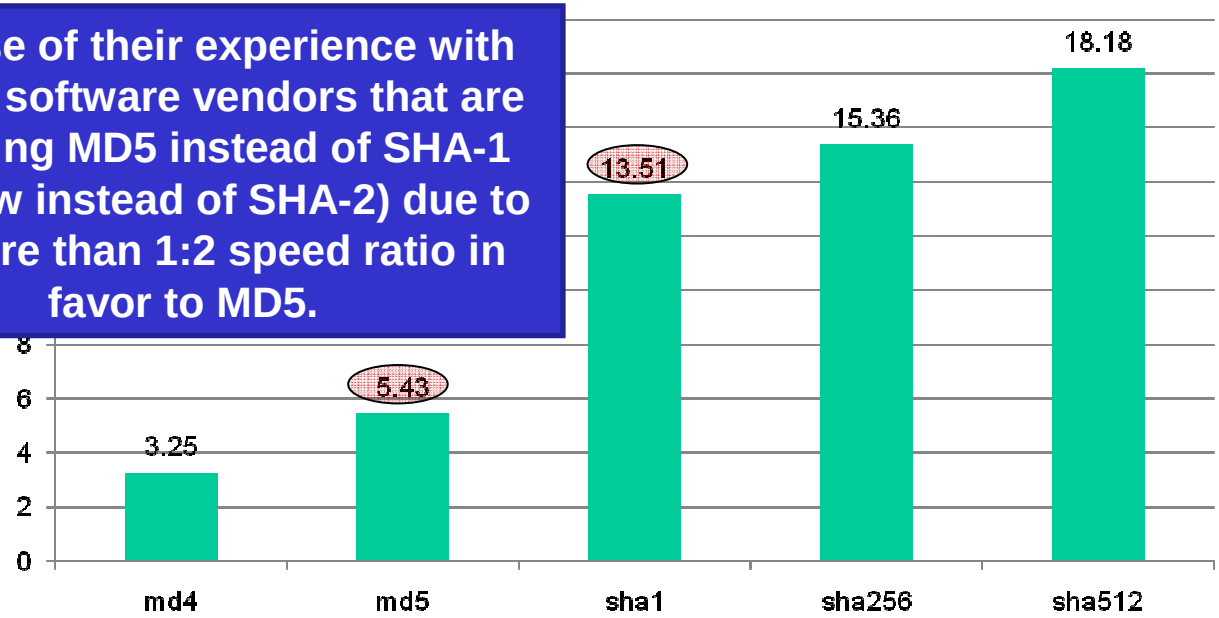

NTNU
Innovation and Creativity

# NIST worldwide hash competition for SHA-3

- Why NIST expects SHA-3 to be “with significantly improved efficiency” (over SHA-2)?

**32-bit Mode, 2400MHz, Intel Core 2 Duo, Cycles/byte, eBASH, supercop-20100120, cobra**

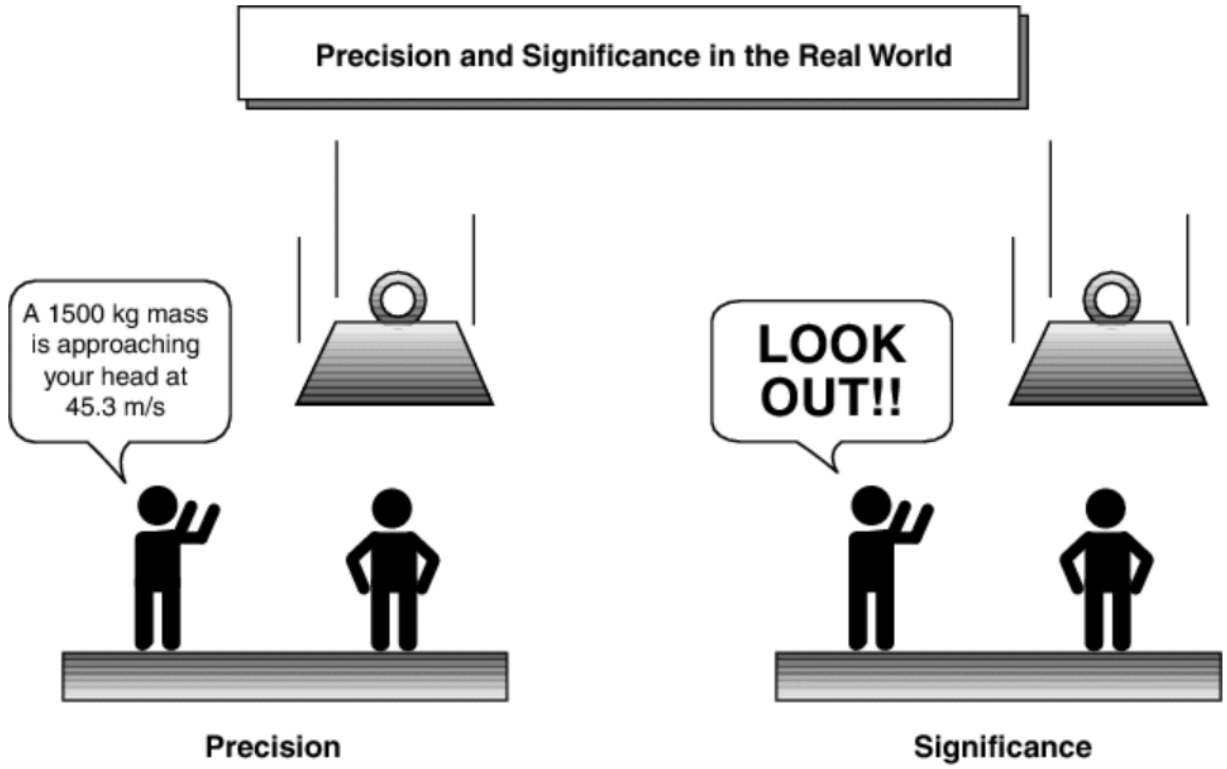
**Because of their experience with serious software vendors that are still using MD5 instead of SHA-1 (and now instead of SHA-2) due to the more than 1:2 speed ratio in favor to MD5.**




NTNU
Innovation and Creativity

# NIST worldwide hash competition for SHA-3

- What does it mean “significantly improved efficiency” (over SHA-2)?
- No clear definition.
- Fuzzy definition?



# NIST worldwide hash competition for SHA-3

- What does it mean “significantly improved efficiency” (over SHA-2)?
- Let us take the following “definition”: Significantly improved efficiency over SHA-2 means at least 2 times faster than SHA-2.
- Then, if a function is 2 times slower than SHA-2, it means that it has significantly worse efficiency.
- This convention seems to be accepted by many in cryptography (see “Classication of the SHA-3 Candidates”, Fleischmann, Forler, and Gorski, eprint 2008, report 511)

# Efficiency of 14 SHA-3 candidates

**32-bit Mode, 2400MHz, Intel Core 2 Duo,**
**Cycles/byte, eBASH, supercop-20100120, cobra**

**significantly more efficient than SHA-2**
**SHA-2**
**slower than SHA-2**
**significantly worse than SHA-2**

| | 32-bit mode, 256 bit hash | Speed cycles/byte |
|---|---|---|
| 1 | Blue Midnight Wish | 7.83 |
| 2 | Shabal | 9.20 |
| 3 | BLAKE | 9.87 |
| 4 | SIMD | 11.45 |
| 5 | CubeHash16/32 | 13.42 |
| 6 | Luffa | 13.90 |
| 7 | SHA-256 | 15.36 |
| 8 | JH | 19.27 |
| 9 | Keccak20 | 20.20 |
| 10 | Fugue | 20.48 |
| 11 | Grøstl | 22.54 |
| 12 | SHAvite-3 | 25.44 |
| 13 | Hamsi | 32.72 |
| 14 | ECHO | 34.23 |
| 15 | Skein | 35.34 |

| | 32-bit mode, 512 bit hash | Speed cycles/byte |
|---|---|---|
| 1 | Blue Midnight Wish | 4.80 |
| 2 | Shabal | 9.20 |
| 3 | SIMD | 12.86 |
| 4 | CubeHash16/32 | 13.42 |
| 5 | BLAKE | 15.77 |
| 6 | SHA-512 | 18.18 |
| 7 | JH | 19.37 |
| 8 | Keccak20 | 20.20 |
| 9 | Luffa | 25.50 |
| 10 | Hamsi | 32.72 |
| 11 | Skein | 35.34 |
| 12 | Grøstl | 36.61 |
| 13 | ECHO | 59.51 |
| 14 | Fugue | 72.00 |
| 15 | SHAvite-3 | 105.19 |

NTNU
Innovation and Creativity

# Efficiency of 14 SHA-3 candidates

**64-bit Mode, 2400MHz, Intel Core 2 Duo, Cycles/byte, eBASH, supercop-20100120, cobra**

**significantly more efficient than SHA-2**

**SHA-2**

**slower than SHA-2**

**significantly worse than SHA-2**

| | 64-bit mode, 256 bit hash | Speed cycles/byte |
|---|---|---|
| 1 | Skein | 6.47 |
| 2 | Shabal | 7.14 |
| 3 | Blue Midnight Wish | 7.41 |
| 4 | BLAKE | 9.86 |
| 5 | SIMD | 10.97 |
| 6 | CubeHash16/32 | 12.67 |
| 7 | Keccak20 | 13.00 |
| 8 | Luffa | 13.40 |
| 9 | SHA-256 | 15.56 |
| 10 | JH | 16.97 |
| 11 | Grøstl | 21.35 |
| 12 | Fugue | 21.45 |
| 13 | SHAvite-3 | 22.31 |
| 14 | ECHO | 29.51 |
| 15 | Hamsi | 31.99 |

| | 64-bit mode, 512 bit hash | Speed cycles/byte |
|---|---|---|
| 1 | Blue Midnight Wish | 3.84 |
| 2 | Skein | 6.47 |
| 3 | Shabal | 7.14 |
| 4 | BLAKE | 9.39 |
| 5 | SHA-512 | 11.54 |
| 6 | SIMD | 12.13 |
| 7 | CubeHash16/32 | 12.67 |
| 8 | Keccak20 | 13.00 |
| 9 | JH | 17.06 |
| 10 | Grøstl | 20.05 |
| 11 | Luffa | 23.20 |
| 12 | Hamsi | 31.99 |
| 13 | ECHO | 51.30 |
| 14 | Fugue | 56.00 |
| 15 | SHAvite-3 | 99.63 |

NTNU
Innovation and Creativity

# Efficiency of some of the 14 SHA-3 candidates on 8-bit MCUs

**8-bit AVR implementations by Daniel Otte (256-bit digests + SHA-1 and MD5)**
**http://www.das-labor.org/wiki/AVR-Crypto-Lib/en**

| Name | Language | Size in ROM in Bytes | Use of RAM in Bytes | Use of stack (RAM) in Bytes | Hashsize in Bits | Size of message blocks in Bits | Cycles for Initialisation | Cycles per block | Cycles per byte | Cycles for Finalisation |
|---|---|---|---|---|---|---|---|---|---|---|
| SHA-1 | asm | 1022 | 28 | 170 | 160 | 512 | 221 | 37070 | 579.22 | 37355 |
| CubeHash-256 | C | 1528 | 130 | 61 | 256 | 256 | 1861535 | 186585 | 5830.78 | 2048817 |
| Shabal-256 | asm | 1580 | 188 | 90 | 256 | 512 | 1087 | 13729 | 214.52 | 50905 |
| SHA-256 | asm | 1598 | 40 | 376 | 256 | 512 | 338 | 50165 | 783.83 | 50559 |
| BlueMidnightWish-256 | asm | 1648 | 68 | 246 | 256 | 512 | 555 | 52046 | 813.22 | 104821 |
| MD5 | asm | 1686 | 20 | 117 | 128 | 512 | 72 | 18297 | 285.89 | 18636 |
| Skein-256-256 | asm | 2052 | 50 | 194 | 256 | 256 | 39951 | 38646 | 1207.69 | 78024 |
| Groestl-256 | C | 2234 | 68 | 250 | 256 | 512 | 363 | 522414 | 8162.72 | 783732 |
| Keccak-256 | C | 3705 | 206 | 447 | 256 | 1088 | 1089 | 1307866 | 9616.66 | 1309070 |
| ECHO-256 | C | 4378 | 90 | 585 | 256 | 1536 | 494 | 222594 | 1159.34 | 224245 |
| Blake-32 | C | 7422 | 53 | 194 | 256 | 512 | 384 | 71362 | 1115.03 | 71961 |

**The goal is the smaller size!**

**Well, the speed should not be too slow!**

NTNU
Innovation and Creativity

# Modes of operations for hash functions

- Analogy with AES competition
  - Before the end of the AES competition, NIST organized workshop for modes of operations of the block ciphers
- Do we need standardized modes of operation for the hash functions?
  - **Multi-core processors**
  - **Multi-core graphic cards**
  - **Incremental hashing**
- **Certainly yes!**

# Outline

- Swiss army knife in cryptography
- What is cryptographic hash function?
- Conflict between theory and practice
- Random oracle paradigm
- Formal definition of cryptographic hash functions and basic attacks
- Choosing the length of hash functions
- Merkle-Damgård iterative construction of hash functions
- A short history of practical hash functions constructions (and their failures)
- Merkle-Damgård Construction does not act as Random Oracle
- Constructions of iterative hash functions that do not suffer from attacks on MD
- Building blocks in the hash functions
- Description of the main applications of hash functions
- NIST worldwide hash competition for SHA-3
- **Advertisment: Blue Midnight Wish Cryptographic Hash Function**

- Advertisement:
- BLUE MIDNIGHT WISH team:
  - Danilo Gligoroski – designer
  - Vlastimil Klima - designer
  - Svein Johan Knapskog (coordinating the synergy in the team, general comments and suggestions for improvements, proofreading)
  - Mohamed El-Hadedy (VHDL implementation)
  - Jørn Amundsen (Big-endian and endian-neutral implementation, suggestions for improvements)
  - Stig Frode Mjølsnes (contributed to an 8-bit implementation)

# What is the status of Blue Midnight Wish so far in SHA-3 competition?

- No mayor attacks on the hash function
- Distinguisher attacks on the compression function

| | Distinguisher of Aumasson | | Distinguisher of Guo and Thomsen | | | |
|---|---|---|---|---|---|---|
| BMW-256 | Compression function 2/14 | BMW-256 | Compression function 0/16 | Compression function 1/15 | Compression function 2/14 | BMW-256 |
| Distinguished bits | 4 | 0 | 9 | 1 | 1 | 0 |
| Distinguished Variables | $H_{new_0}$ | / | $H_{new_0}$, $H_{new_5}$ | $H_{new_0}$ | $H_{new_0}$ | / |
| $SecurityMargin$ | 98.44% | 100% | 96.48% | 99.61% | 99.61% | 100% |
| BMW-512 | Compression function 2/14 | BMW-512 | Compression function 0/16 | Compression function 1/15 | Compression function 2/14 | BMW-512 |
| Distinguished bits | 4 | 0 | 11 | 1 | 1 | 0 |
| Distinguished Variables | $H_{new_0}$ | / | $H_{new_0}$, $H_{new_5}$ | $H_{new_0}$ | $H_{new_0}$ | / |
| $SecurityMargin$ | 99.22% | 100% | 97.85% | 99.80% | 99.80% | 100% |

Table 2: A summary of the distinguished bits in BLUE MIDNIGHT WISH and their reflections to the security margins both for the whole hash function as well as for the compression function

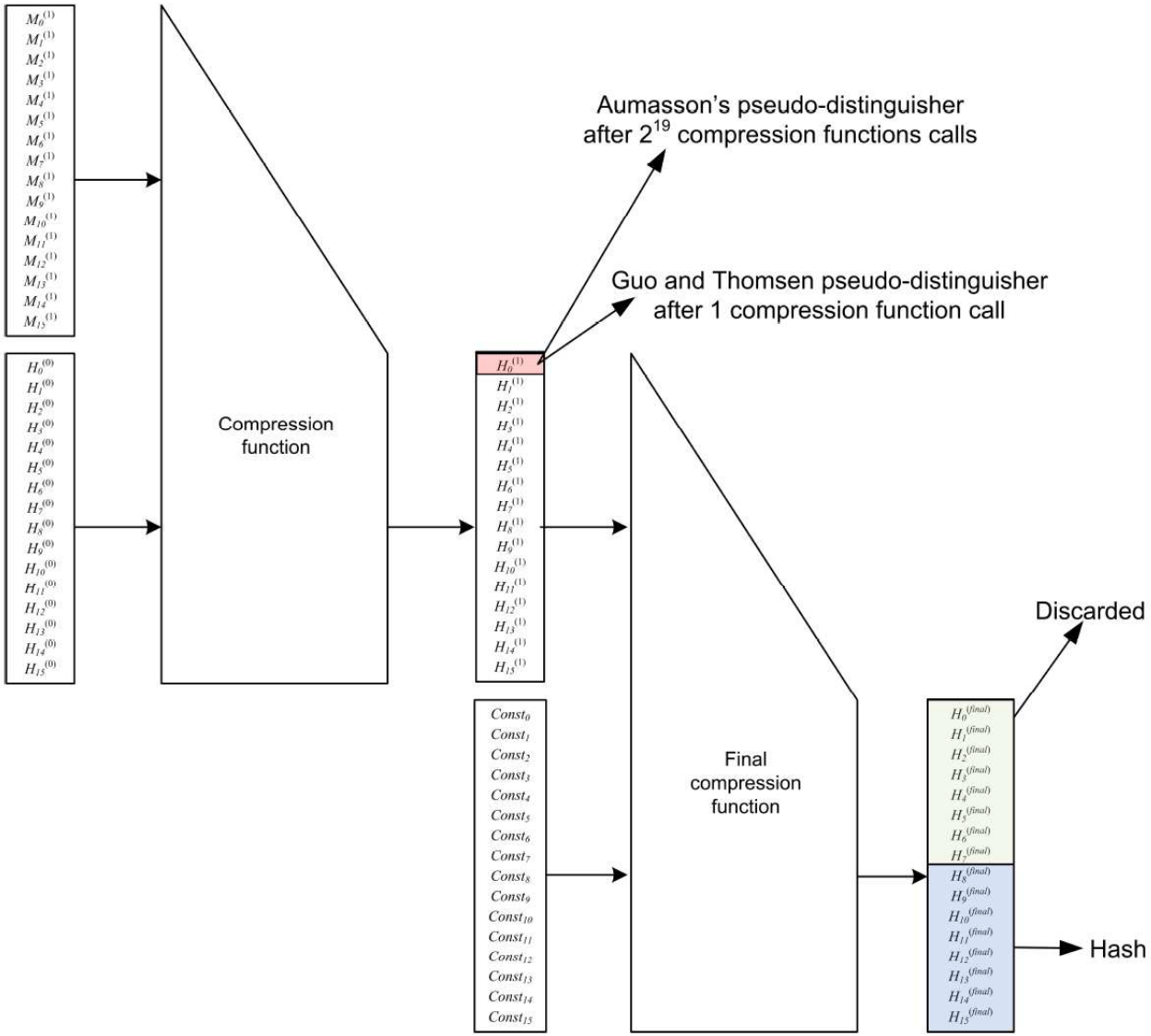



Figure 1: A graphic presentation of a complete hashing of a message consisting of one block and the connection with the distinguishing attacks (in this context pseudo-distinguishing attacks) on the BLUE MIDNIGHT WISH hash function.

- Advertisment
- The compression function of Blue Midnight Wish is highly paralelizable
- In new Itaniums (Tukwila) the performance is ~2 cycles/byte
- Soon expect new SSSE3 code for BMW256 (to regain the Nr. 1 position in the race ☺ )
- Speculation: if implemented in a simmilar fashion as AES-NI in Intel CPUs the speed can be less than 0.15 cycles/byte

# NIST SHA-3 hash competition

**This year on August 23-24, 2010 NIST will held Second Hash Workshop**

**Shortly after that they will announce the 5 finalists of SHA-3 competition**

# Thank you for your attention!