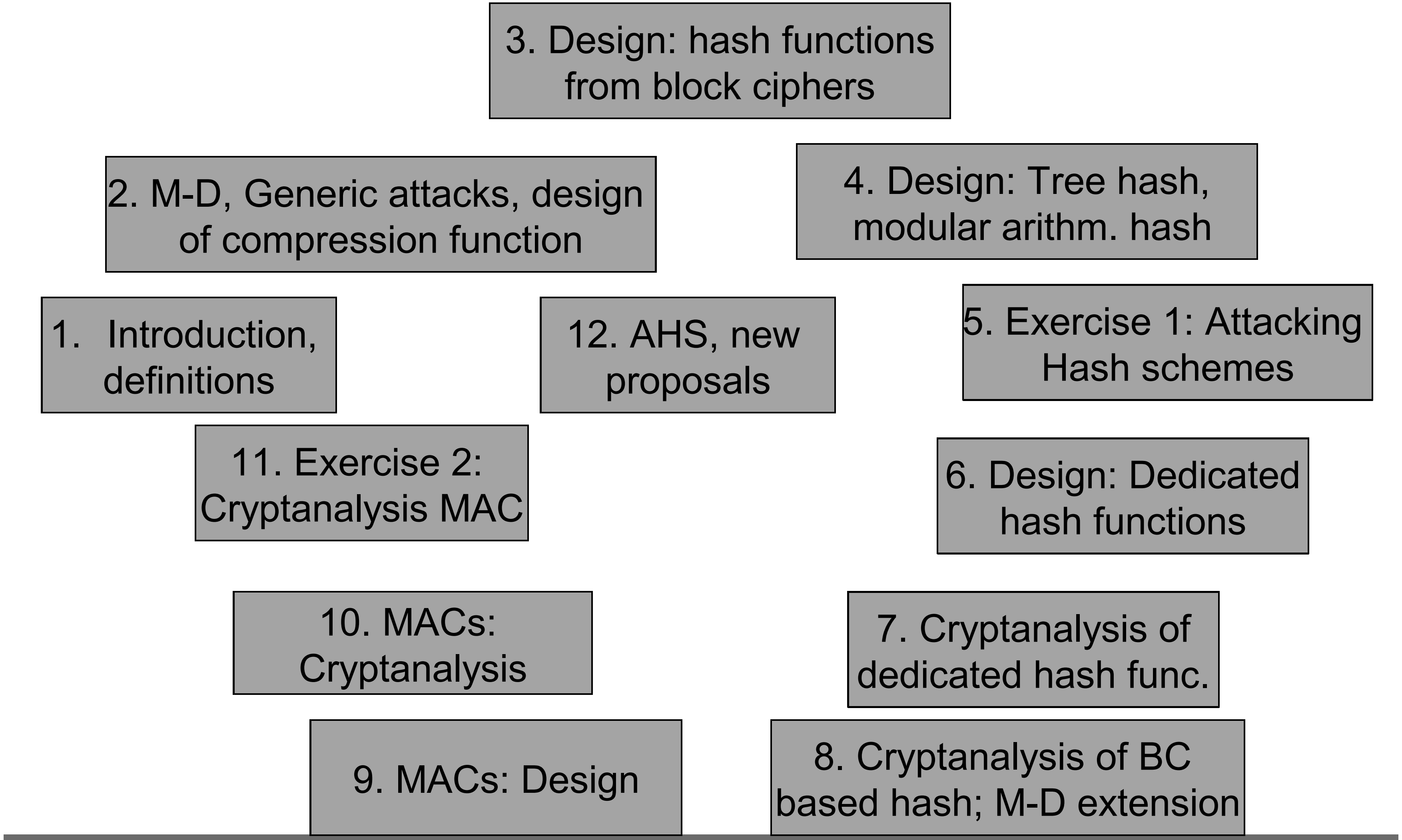
Cryptographic hash functions and MACs Introduction

Lecturer: Enes Pasalic

Scope of the course

- Time schedule: 10 lectures a 2h + 2 class exercises a 2h; 3x a week
- Course level: **Advanced**, suited for graduate students; though undergraduate students are also encouraged
- Exam – Written, date to be announced later
- Literature:
  - "Handbook of applied cryptography", Menezes, Oorschot, Vanstone; Chapters 9 and (10), Bart Preneel, Ph. D Thesis, 1993
  - Scientific articles, references to important ones will be given
- A small project for interested students is an option (more ECTS credits)
- Background: Basic discrete math and probability theory

Course topics



1. Introduction, definitions

2. M-D, Generic attacks, design of compression function

3. Design: hash functions from block ciphers

4. Design: Tree hash, modular arithm. hash

5. Exercise 1: Attacking Hash schemes

6. Design: Dedicated hash functions

7. Cryptanalysis of dedicated hash func.

8. Cryptanalysis of BC based hash; M-D extension

9. MACs: Design

10. MACs: Cryptanalysis

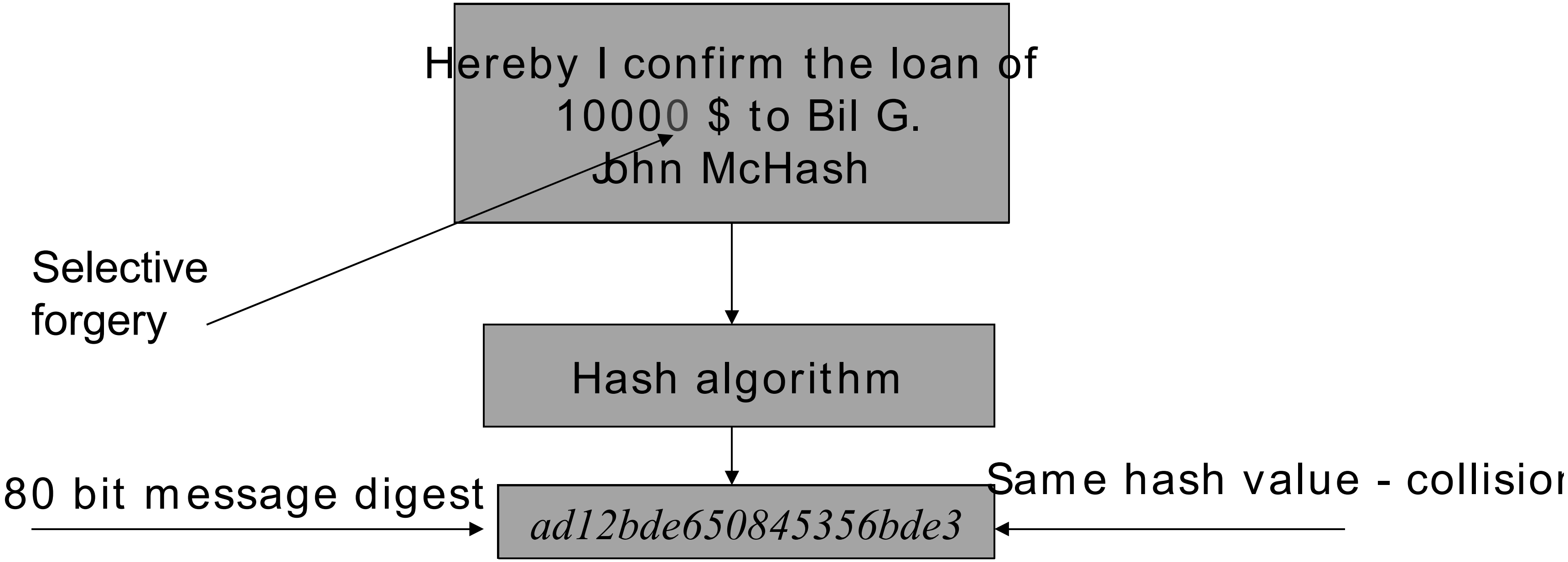
11. Exercise 2: Cryptanalysis MAC

12. AHS, new proposals

Overview of the course

What is a hash function ?

- Informally a message of arbitrary length is mapped to a *hash value, message digest, hash code* of length *n*.
- Formally h: $\{0,1\}^*$ -----> $\{0,1\}^n$,



Hereby I confirm the loan of 10000 $ to Bil G. John McHash

Selective forgery

Hash algorithm

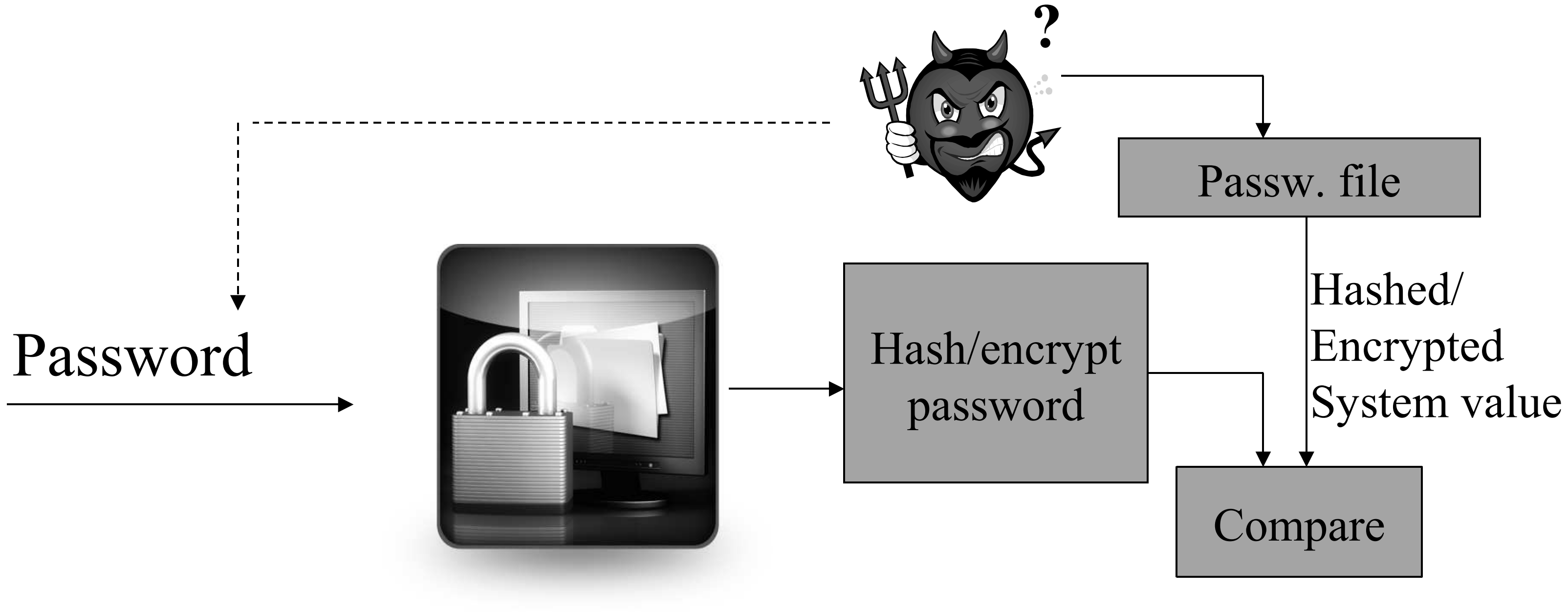
80 bit message digest

ad12bde650845356bde3

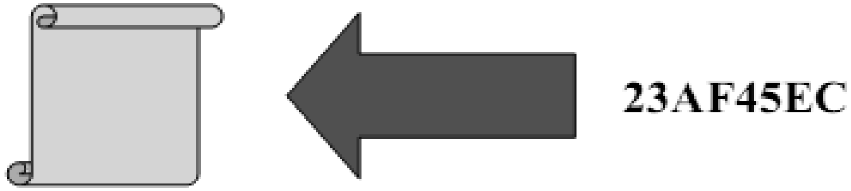
Same hash value - collision

Why do we need hash functions ?

- One familiar application is hashing the password.
- Logging on your computer requires a password – it is saved in encrypted or in hashed form
- Problem given $h(m)$ find m – *one way property*

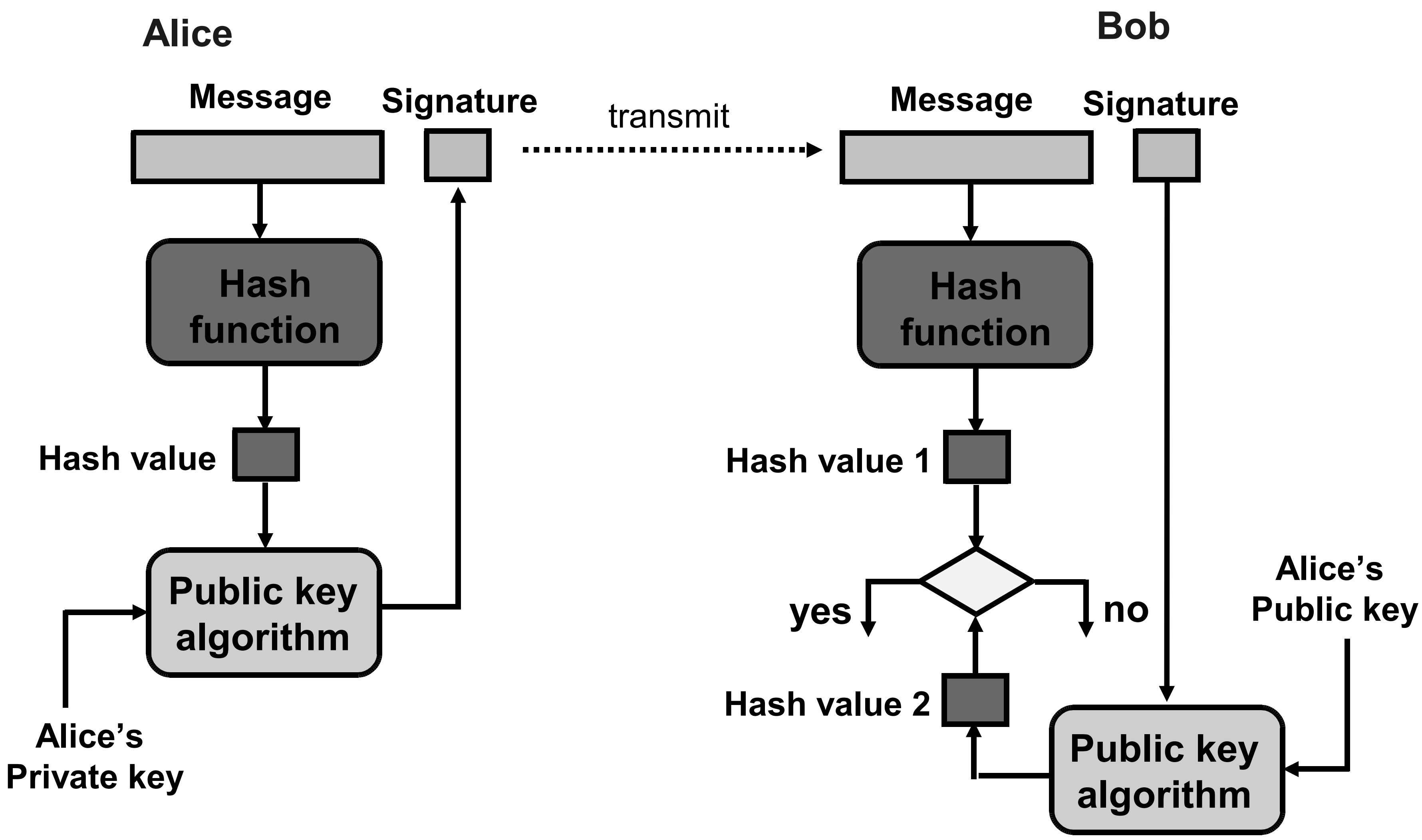


Preimage resistance



- Generic attack:
  - Try out inputs
  - Complexity: 2^n
  - with quantum computer: $2^{n/2}$

Digital Signature



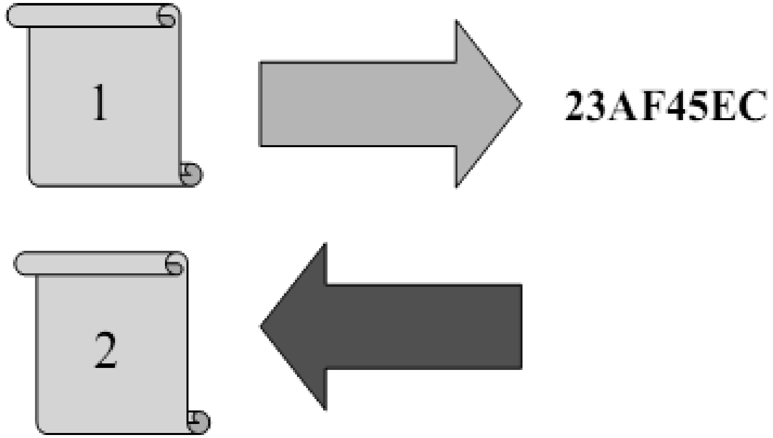
Signing message digest

- The problem is that public key algorithms are slow, exponentiation in RSA, $m^d \bmod N$ takes $O(N^2)$ clocks. For 1024-bit number 10^6 clocks !!

- If the message is long this becomes a problem – solution is to compress the message by hashing and then sign.

Attacker tries to find collision for a given hash value, e.g. claiming he signed another message

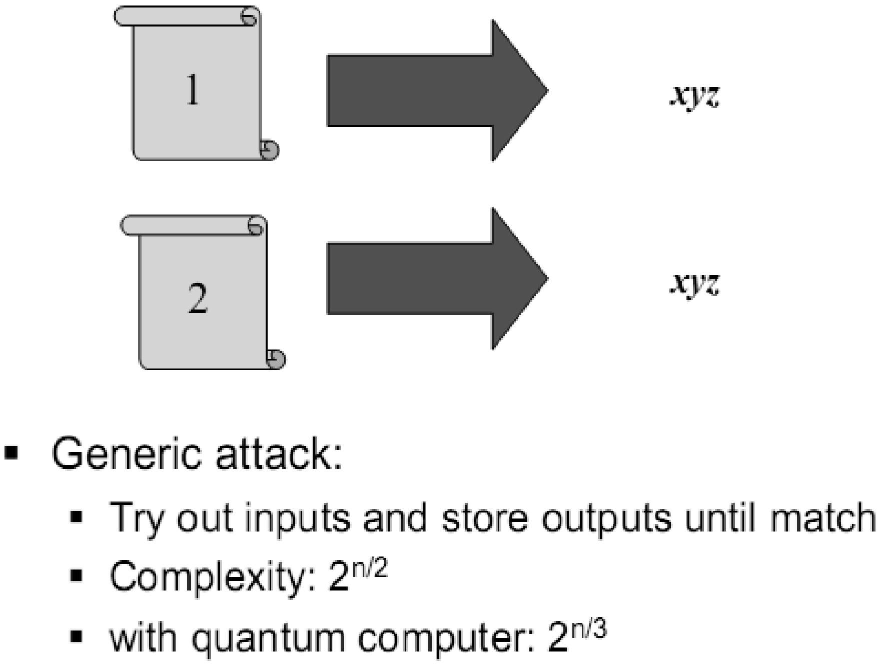
Second preimage resistance



- Generic attack:
  - Try out inputs different from input 1
  - Complexity: 2^n
  - with quantum computer: $2^{n/2}$

Collision resistance

- Usually only existential forgery (not in control of the messages), but indicates the weakness of hash design



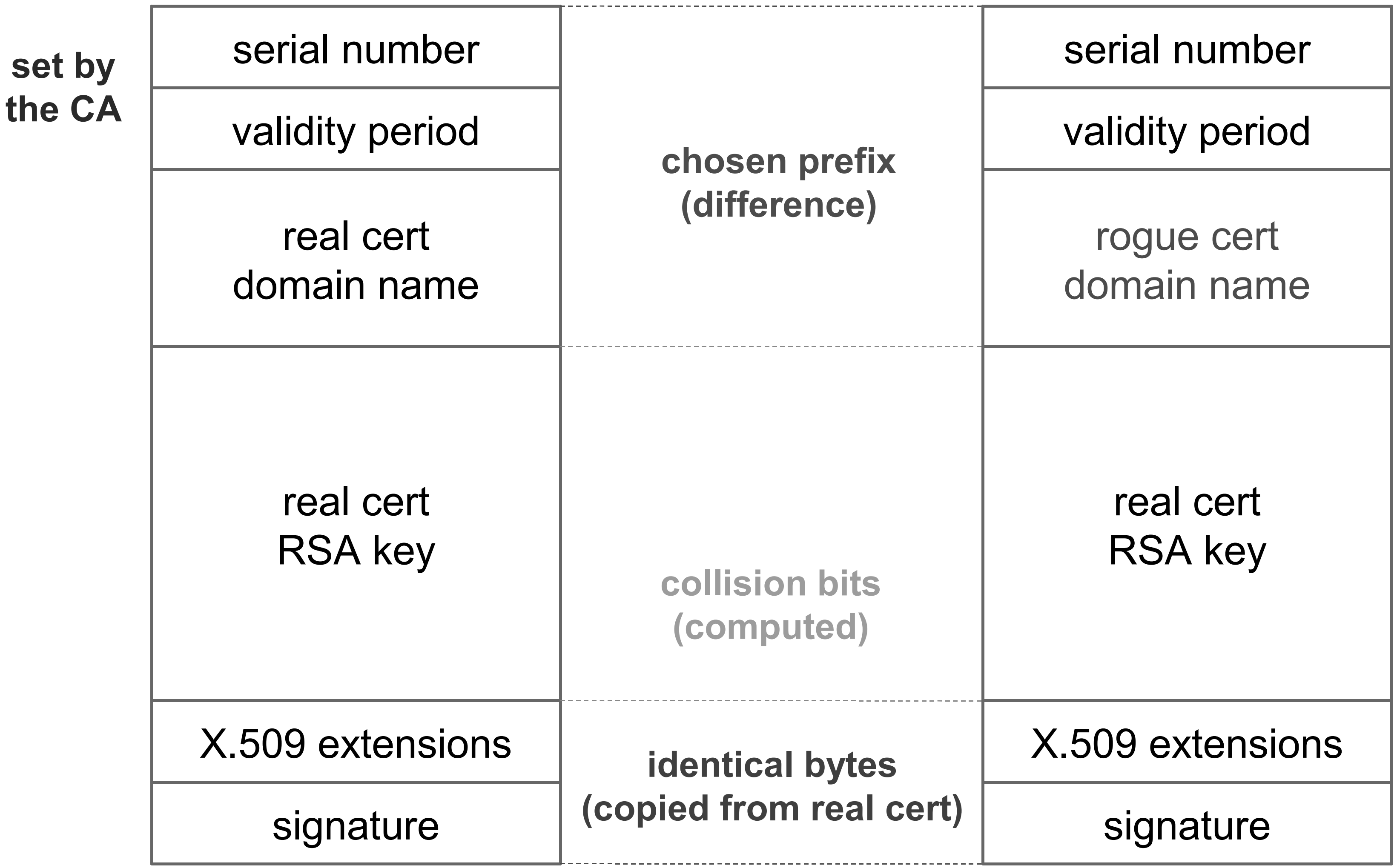
- Generic attack:
  - Try out inputs and store outputs until match
  - Complexity: $2^{n/2}$
  - with quantum computer: $2^{n/3}$

Who cares if we get collisions for two random messages ?

- Can we find meaningful collisions ?

- $2^{n/2}$ due to birthday paradox

X.509 certificates



set by the CA

serial number

validity period

real cert domain name

real cert RSA key

X.509 extensions

signature

chosen prefix (difference)

collision bits (computed)

identical bytes (copied from real cert)

serial number

validity period

rogue cert domain name

real cert RSA key

X.509 extensions

signature

Meaningful coliding fields in X.509

```
------------------------------------------- | -----------------------------------------------------
 30 54                                      | subject distinguished name starts here
 31 19      15                              |
 30 17      13                              |
 06 03      550403                          |
                                            | subject common name:
 13 10      41726A656E204B2E204C656E73747261 | (``Arjen K. Lenstra'')
 13 0C      4D6172632053746576656E73         | (``Marc Stevens'')
 31 16      1A                              |
 30 14      18                              |
 06 03      55040A                          |
                                            | subject organization
 13 0D      436F6C6C6973696F6E61697273       | (``Collisionairs'')
 13 11      436F6C6C6973696F6E20466163746F72 | (``Collision Factory'')
            79                              | (dummy text, used to fill up to convenient byte size)
 31 12                                      |
 30 10                                      |
 06 03      550407                          |
 13 09      45696E64686F76656E              | subject locality (``Eindhoven'')
 31 0B                                      |
 30 09                                      |
 06 03      550406                          |
 13 02      4E4C                            | subject country code (``NL'')
 30 820422                                  |
------------------------------------------- | -----------------------------------------------------
```

Historical development

- R. Merkle introduced the concepts of one-way functions, preimage and 2-nd preimage resistance, tree authentication late 70s
- Universal classes of hash functions– Carter & Wegman, late 70s
- Simmons, authentication codes, late 70s
- Hash functions based on block ciphers, late 70s
- Damgård - CRHF (Collision Resistant Hash Function) late 80s
- MDx and SHA families from mid 90s (dedicated hash functions)
- Hash functions based on modular arithmetic

Design methods for hash functions

- There are four main construction methods for hash functions:

  1. Hash functions based on block ciphers
  2. Customized hash functions
  3. Hash functions based on modular arithmetic
  4. Provably secure hash functions based on number theory e.g. using DiscreteLog Problem

- Also non-sequential approach – tree hashing

Iterative hashing

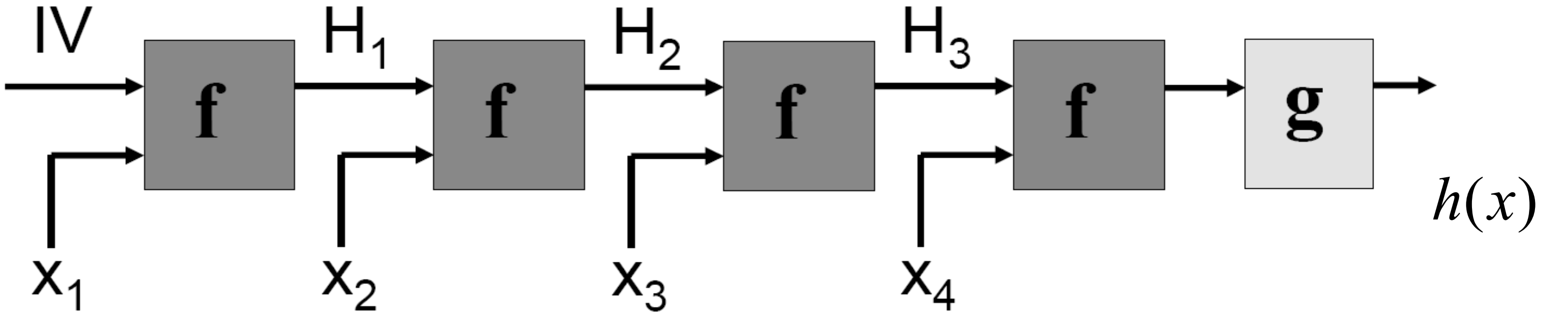
- Since hash functions maps arbitrary length to a fixed length **obvious choice is iterative processing of the message.**

- To compute a hash value of message M, M is split into blocks of fixed length $M=M_1 || M_2 || \ldots || M_t$ and each block is processed in a similar way.

- Need for a compression function $f: \{0,1\}^{n+r}$ ---> $\{0,1\}^{n}$

$$
\begin{aligned}
&H_0 = IV \\
&H_i = f(H_{i-1}, M_i) \quad i = 1 \ldots t \\
&h(M) = H_t
\end{aligned}
$$

Merkle-Damgård chaining



- Easy and elegant **but many problems**: for instance remove x_1 and use H_1 instead of IV (if IV is not fixed)

Iterated hashing – general model

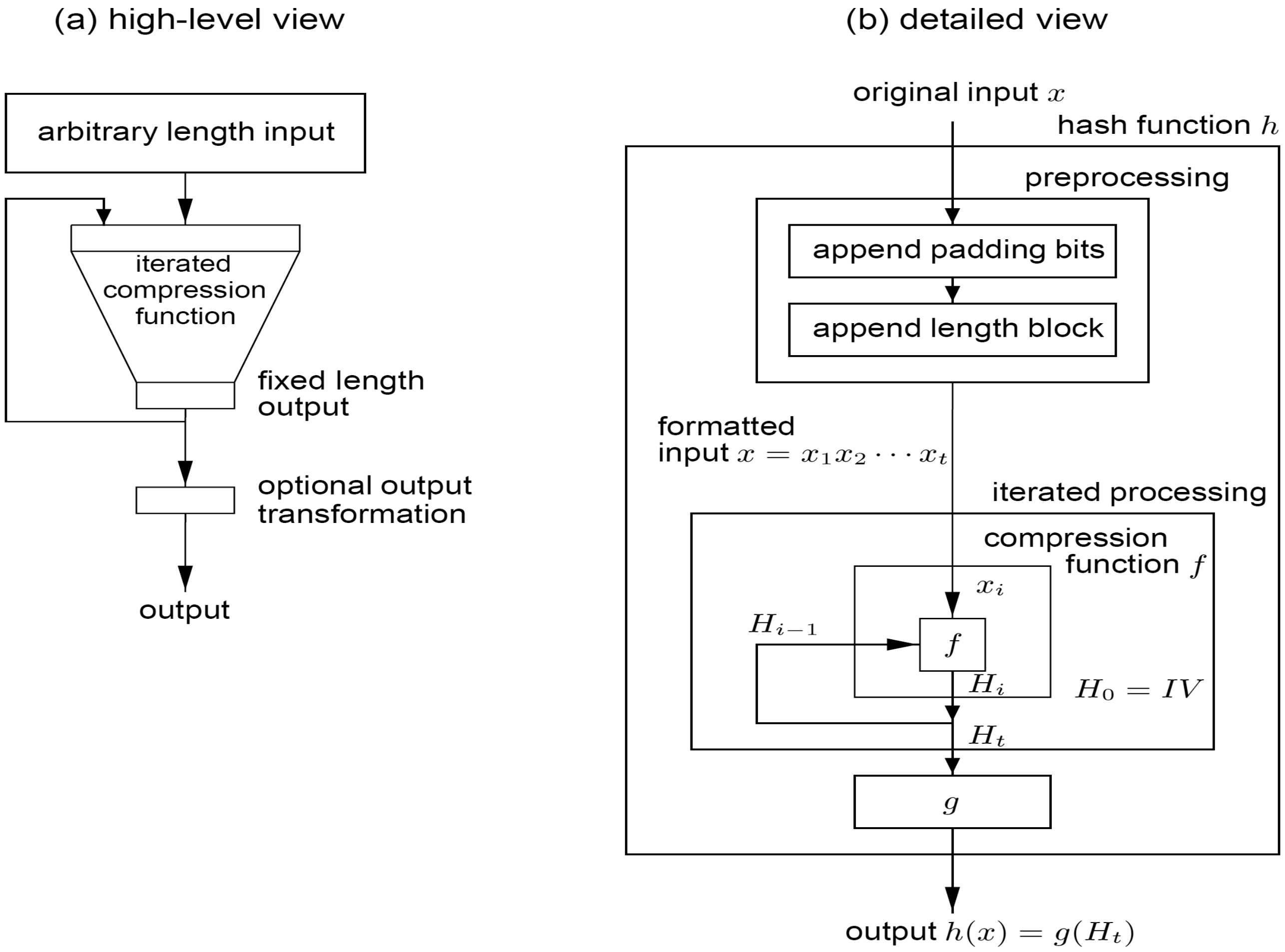
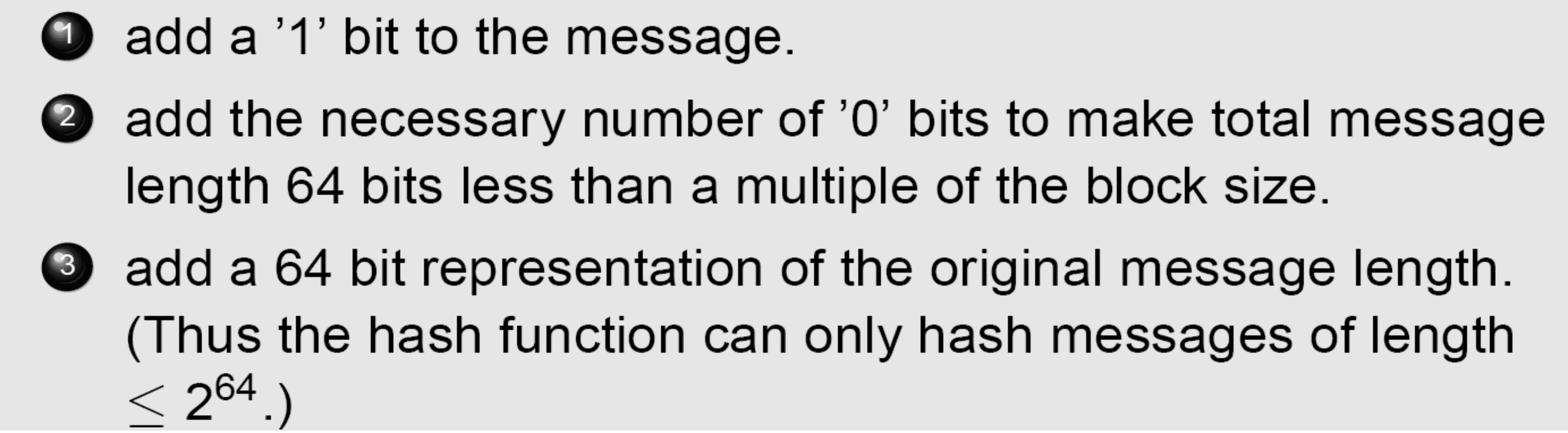


Figure 9.2: *General model for an iterated hash function.*

Merkle-Damgård meta method

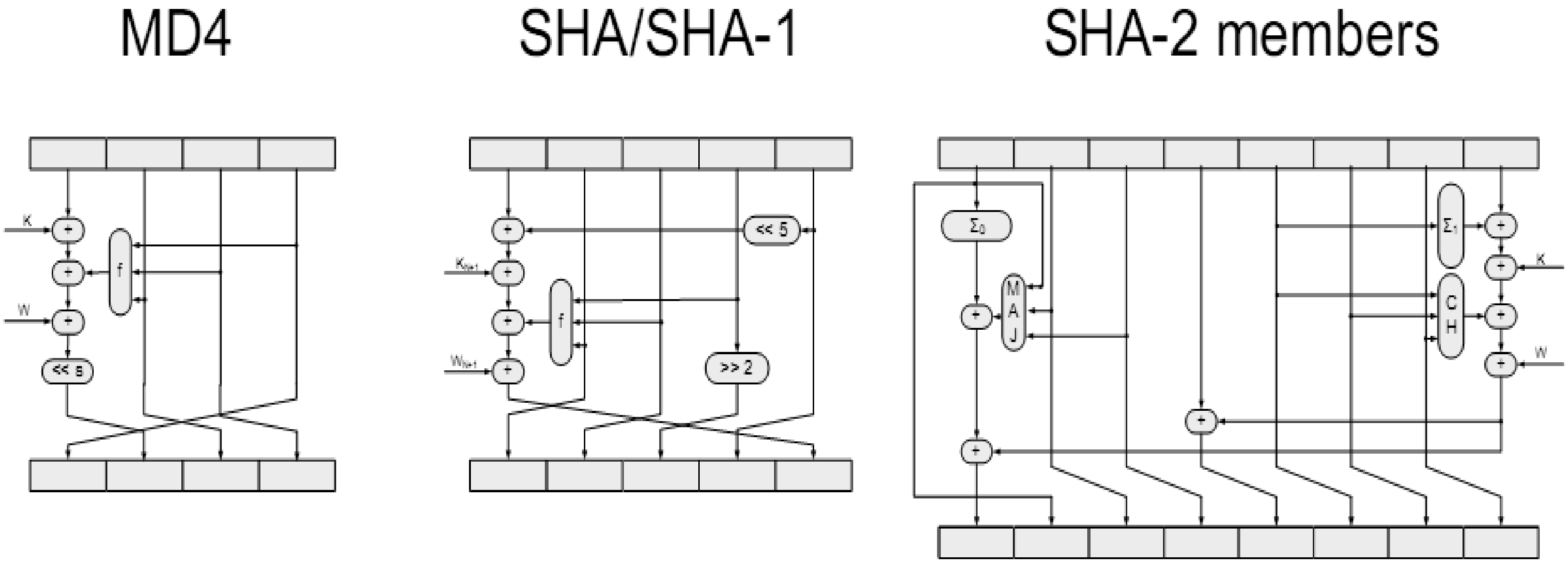
- Merkle-Damgard method prevents that there is a message which is a tail of another message.



1. add a ’1’ bit to the message.
2. add the necessary number of ’0’ bits to make total message length 64 bits less than a multiple of the block size.
3. add a 64 bit representation of the original message length. (Thus the hash function can only hash messages of length $\leq 2^{64}$.)

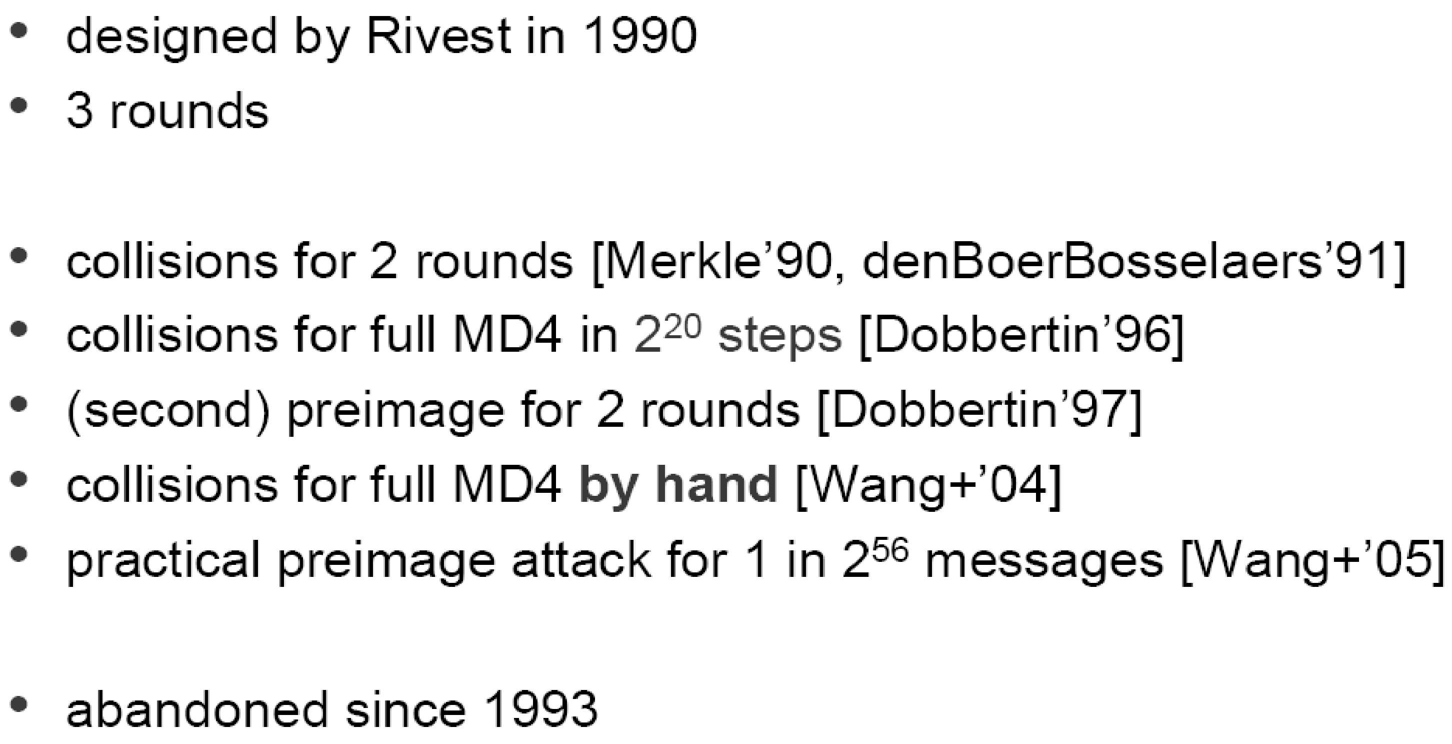
- It remains to find collision resistant compression function or one-way compression function !

Dedicated hash functions



Most famous family of hash functions, even new standard AHS changed the name to SHA-3

MD4 and tweaks



- designed by Rivest in 1990
- 3 rounds

- collisions for 2 rounds [Merkle'90, denBoerBosselaers'91]
- collisions for full MD4 in 2^{20} steps [Dobbertin'96]
- (second) preimage for 2 rounds [Dobbertin'97]
- collisions for full MD4 **by hand** [Wang+'04]
- practical preimage attack for 1 in 2^{56} messages [Wang+'05]

- abandoned since 1993

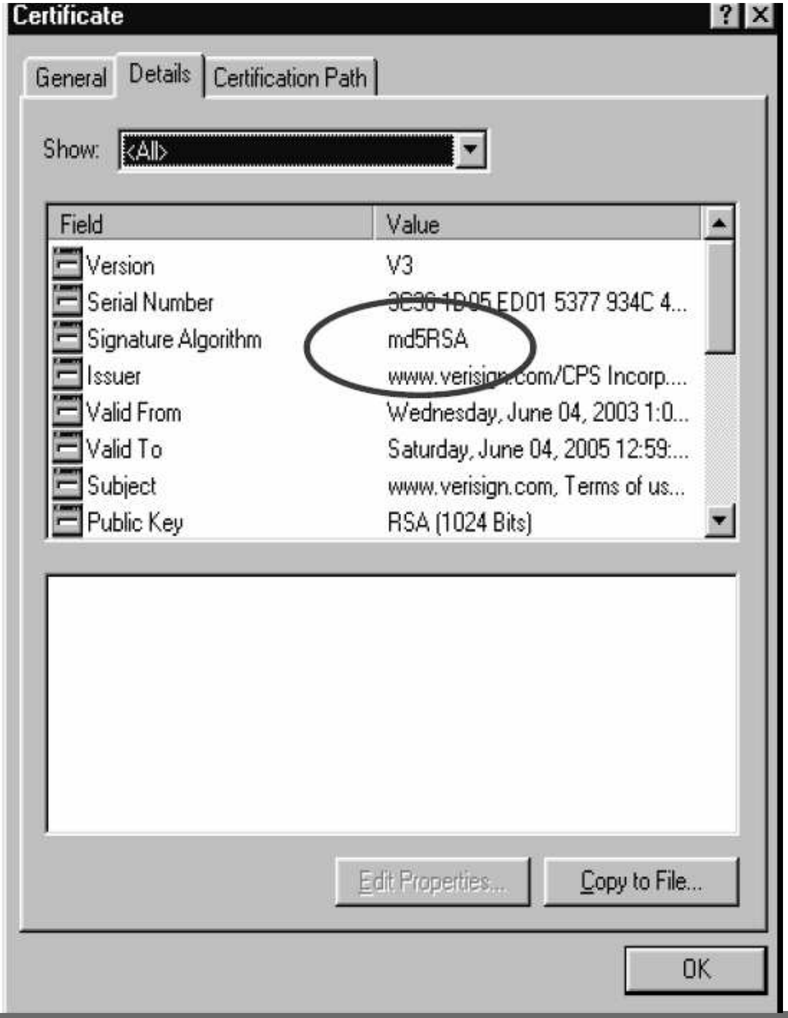
- Replacements and derivatives, MD5, SHA, SHA1, SHA-256 ...

MD5 security

MD5

- Advice (RIPE since ‘92, RSA since ‘96): stop using MD5
- Largely ignored by industry (click on a cert...)

- Collisions for MD5 are within range of a brute force attack anyway (2^{64}): with 100.000$ a few days
- [Wang+’04] collision in 15 minutes on a PC
- 2007: collisions in seconds



Show: <All>

Field	Value
Version	V3
Serial Number	3E36 1D05 ED01 5377 934C 4...
Signature Algorithm	md5RSA
Issuer	www.verisign.com/CPS Incorp....
Valid From	Wednesday, June 04, 2003 1:0...
Valid To	Saturday, June 04, 2005 12:59:...
Subject	www.verisign.com, Terms of us...
Public Key	RSA (1024 Bits)

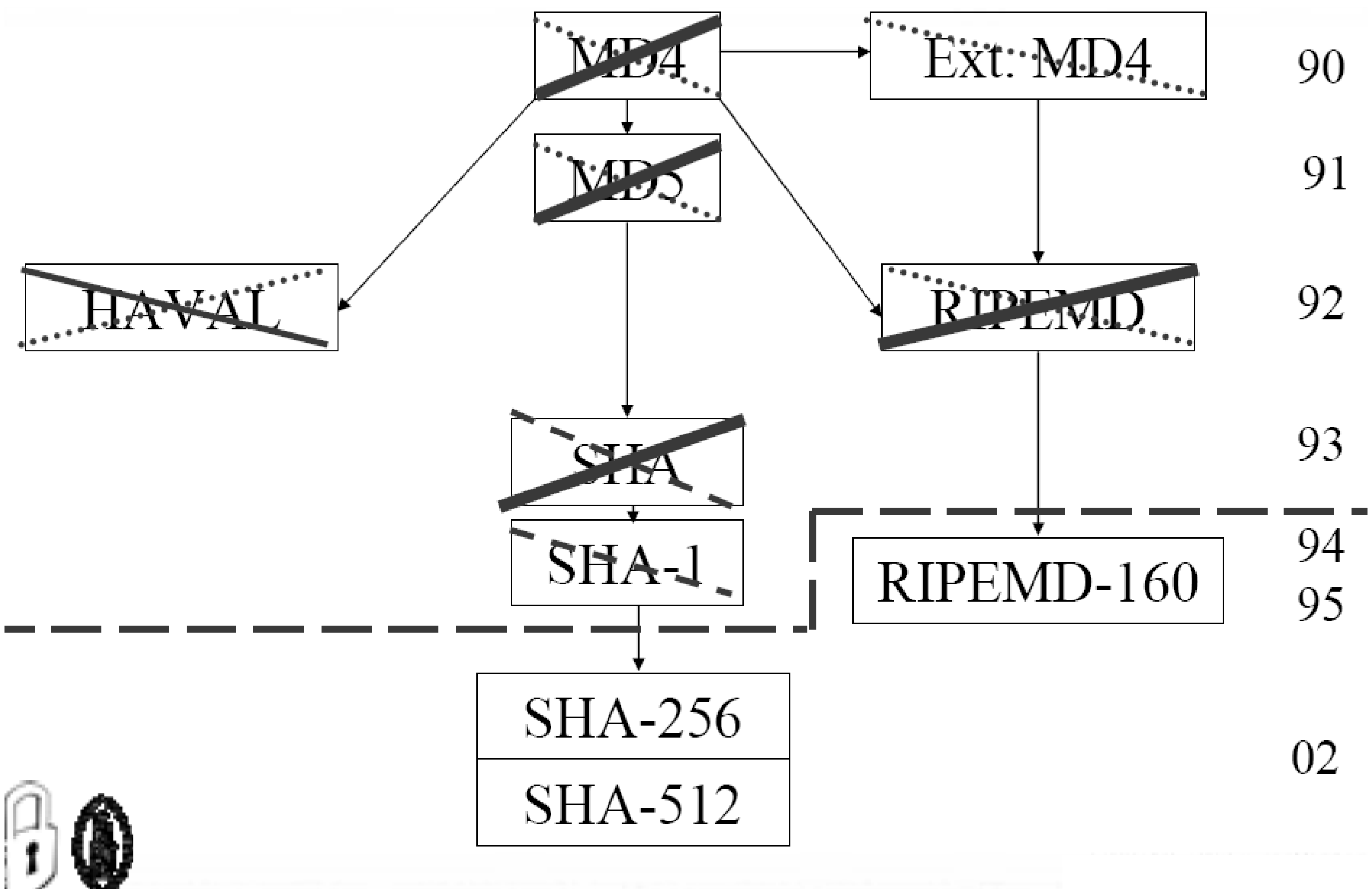
SHA-1

- SHA designed by NIST (NSA) in ‘93
- redesign after 2 years (’95) to SHA-1

- Collisions found for SHA-0 in 2^{51} [Joux+’04]
- Reduced to 2^{39} [Wang+’05] and 2^{32} [Rechberger+’07]

- Collisions for SHA-1 in 2^{63} [Wang+’05]
- Collisions for SHA-1 found for 70 out of 80 rounds [De Cannière-Mendel-Rechberger’07] in 2^{44}

History of MD family

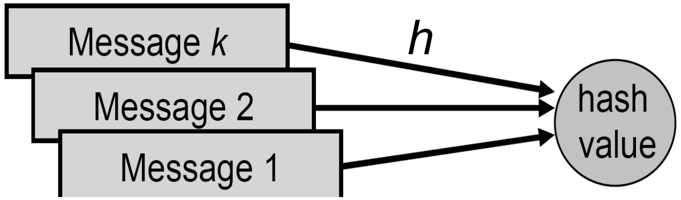


Requirements on hash functions (informal)

- For any message *m*, it is easy to compute *h*(*m*)
- Given *h*(*m*), there is no way (cheaper than brute force) to find a *m* that hashes to *h*(*m*)
- It is computationally impossible to find two different *m* and *m'* which hash to the same value *h*(*m*)

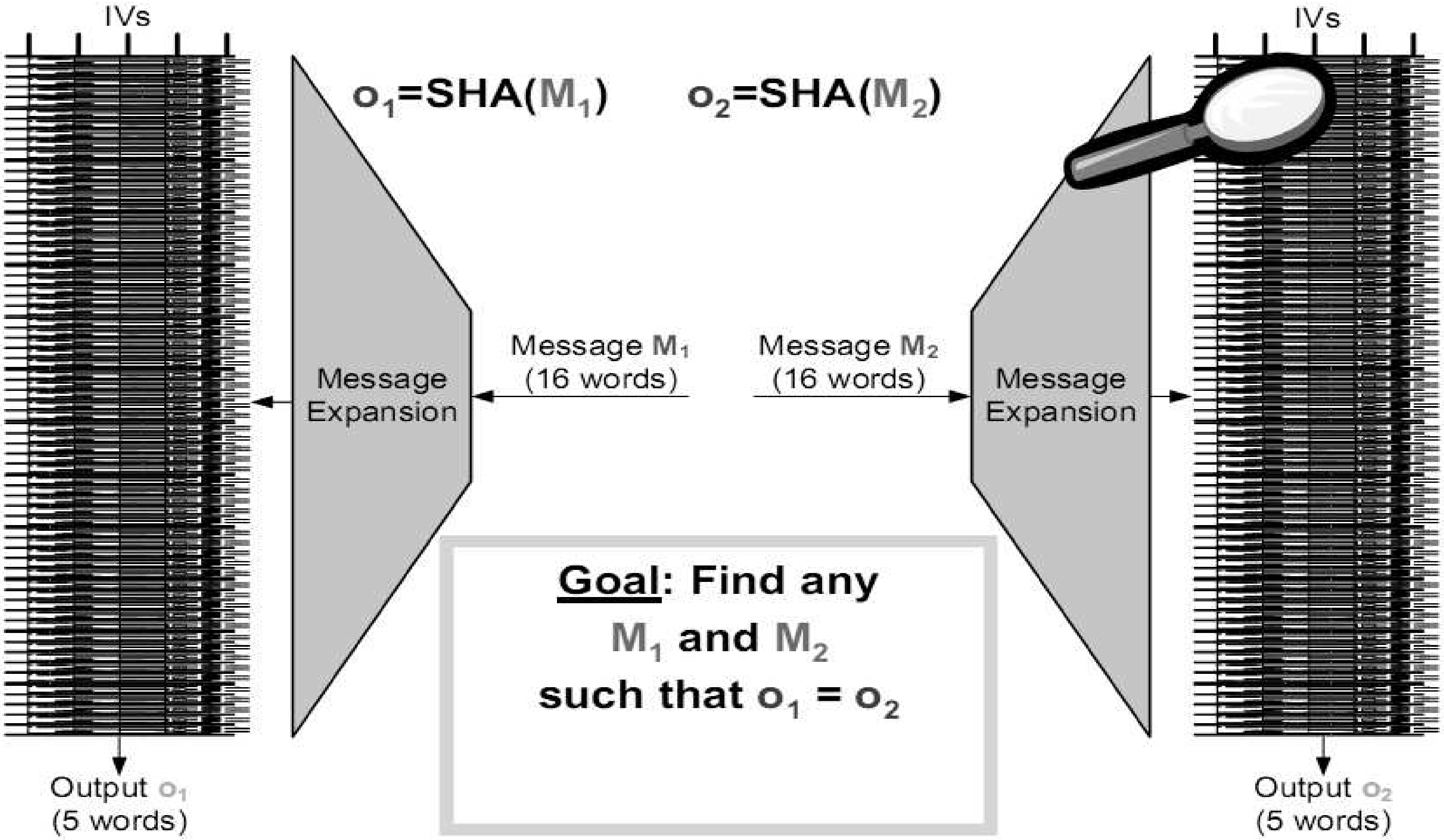
It is necessary for the transformation that the output must not be predictable:

- If 1000 inputs are selected at random, any particular bit in the 1000 resulting outputs should be “1” about half the time
- Each output should have about 50% of “1” bits (with high probability)
- If two inputs differ only by one bit, the outputs should look like independently chosen random numbers



Many messages map to the same hash value

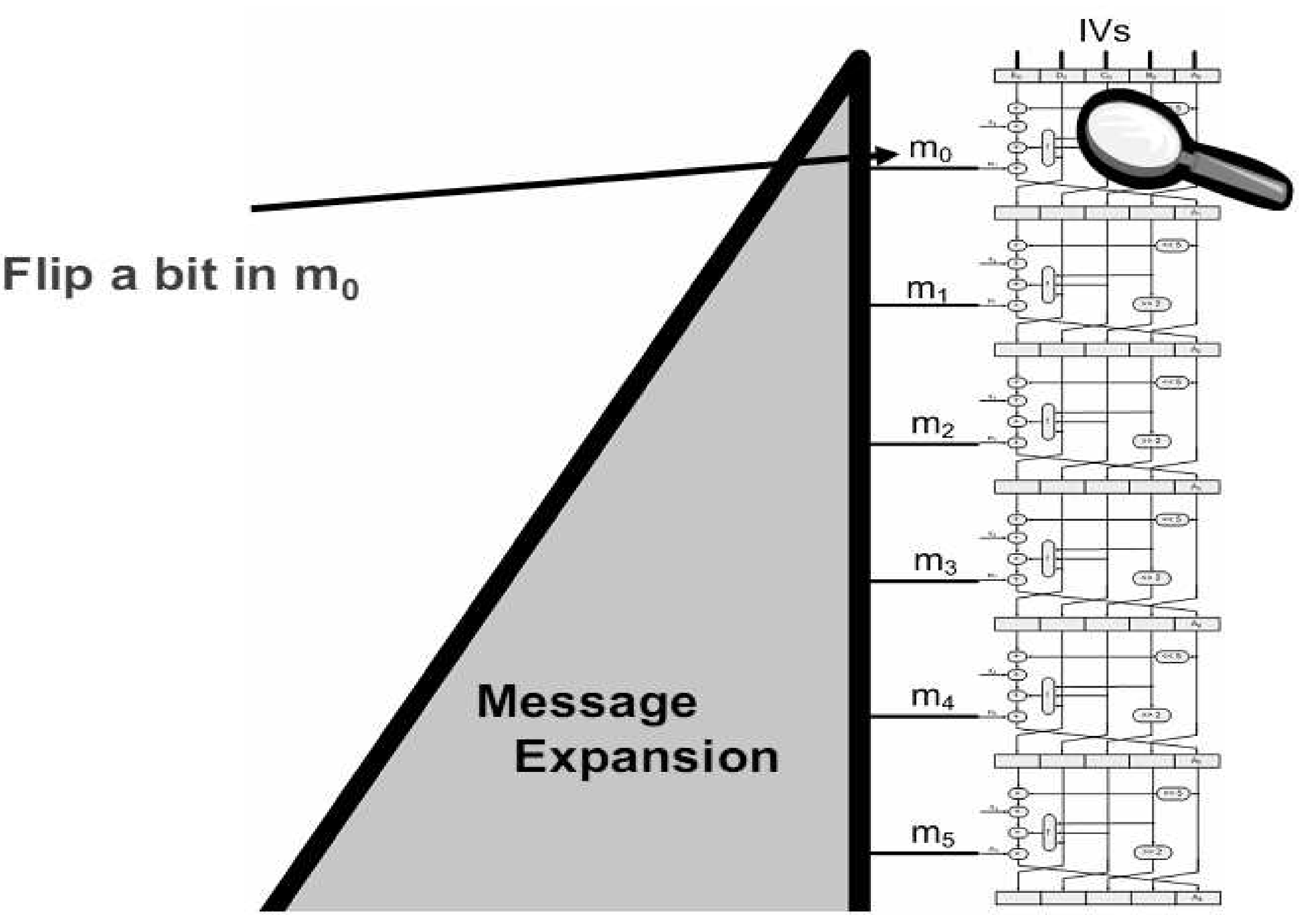
Flipping a single bit – SHA



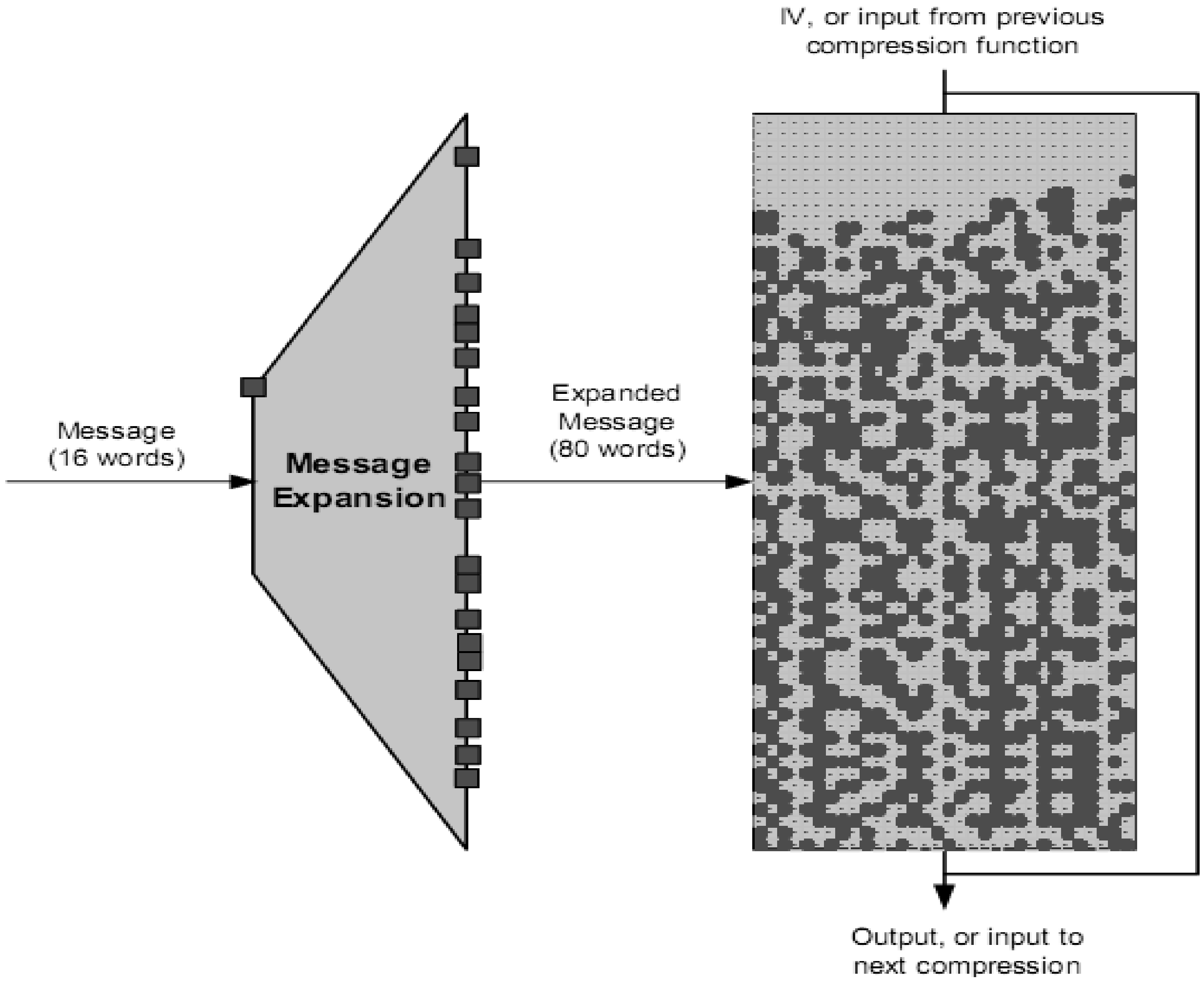
o_1=SHA(M_1) o_2=SHA(M_2)

Goal: Find any M_1 and M_2 such that $o_1 = o_2$

Flipping a single bit II



Effect of flipping a single bit III


IV, or input from previous compression function
Message (16 words)
Message Expansion
Expanded Message (80 words)
Output, or input to next compression

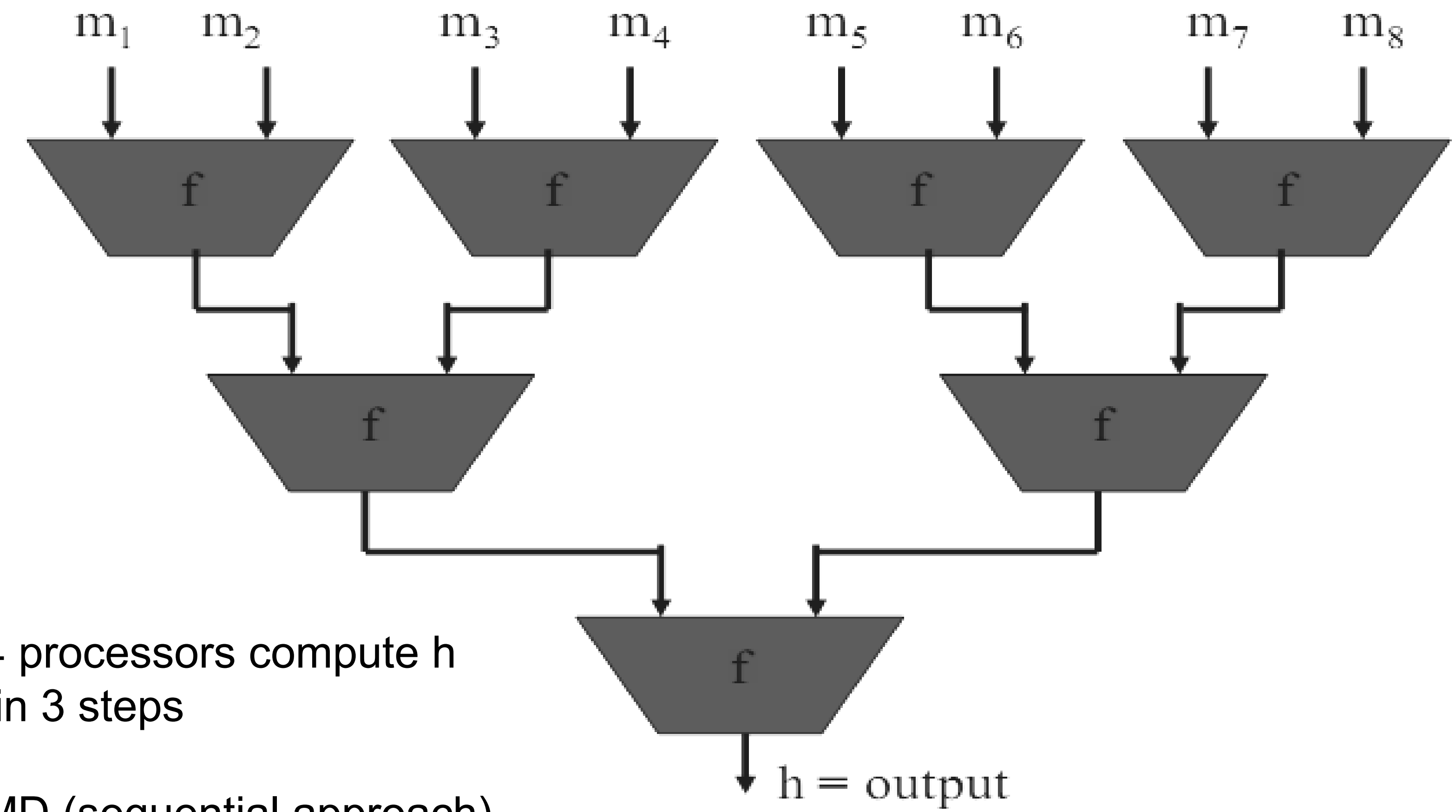
Flipping “a few dedicated bits”

$$\begin{aligned} M_0 &= a16fce6c0d4bae6eb76cdd7e1e08c387\textcircled{b}c4ac82d5206c98418988efcb9ea44fb \\ &\quad b518923a383e775faa6540fa7720e4a8797d5329c8d87bd0e25feda0a1307ff2 \\ M_0' &= \textcircled{2}16fce6c0d4bae\textcircled{2}e\textcircled{3}76cdd7\textcircled{c}1e08c387\textcircled{3}c4ac82dd206c98418988efcb9ea44fb \\ &\quad b518923a383e775faa6540fa7720e4a8797d5329c8d87bd0e25feda0a1307ff2 \end{aligned}$$

Single bit difference b=1011
3=0011

- **Collision for reduced round SHA-1 (58 rounds out of 80), current record 70/80 rounds**

Alternative to MD - Tree approach

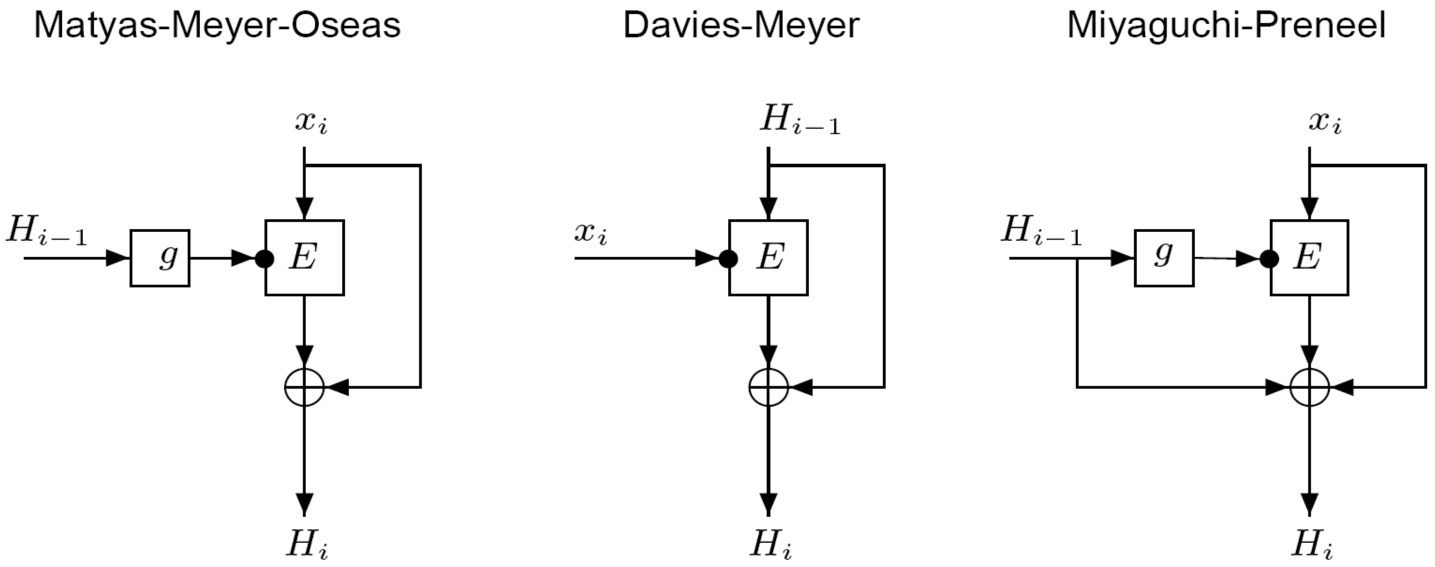


4 processors compute h
in 3 steps

MD (sequential approach)
needs 8 comput.

Secure hash functions from block ciphers

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

Figure 9.3: *Three single-length, rate-one MDCs based on block ciphers.*

Rate of BC hash functions

- Consider DES, the block size is 64 bits and key length 56 bits
- Message digest only 64 bits : hash 2^{32} random messages and find collision (birthday paradox)
- AES : block length 128 bits, key length variable, say 128 bits

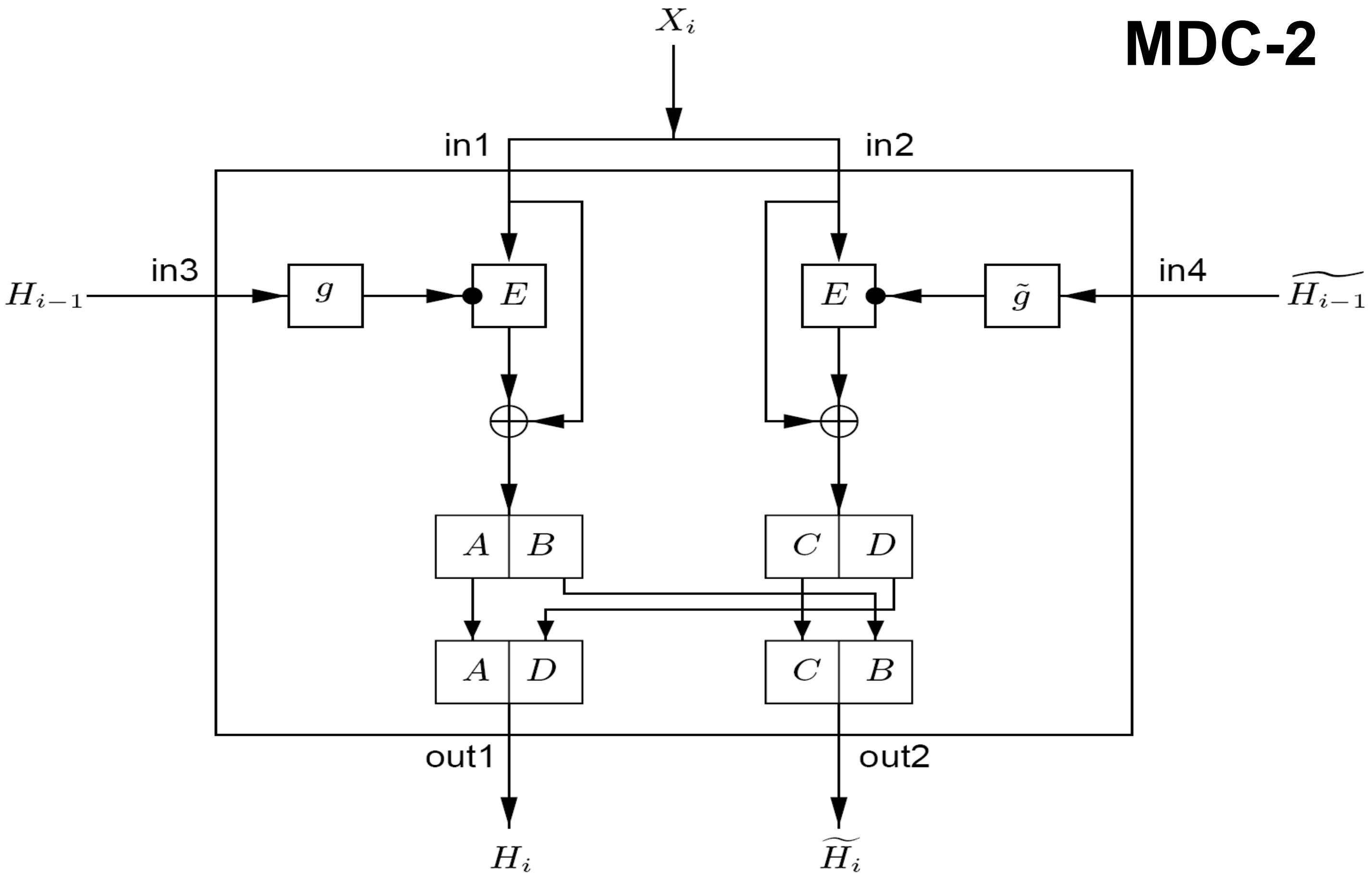
-- Not enough for a long term security hash 2^{64} random messages

- Both of rate 1 – one encryption per message block

Solution – double-length hash functions

Rate 1/2 double-length compression

MDC-2



Why we do not use BC based hashing ?

Main reasons :

- One or more encryptions to process a single block (still slow)
- Key schedule for each encryption

- Security of underlying cipher does not necessarily imply security of hash function, due to iterated structure.

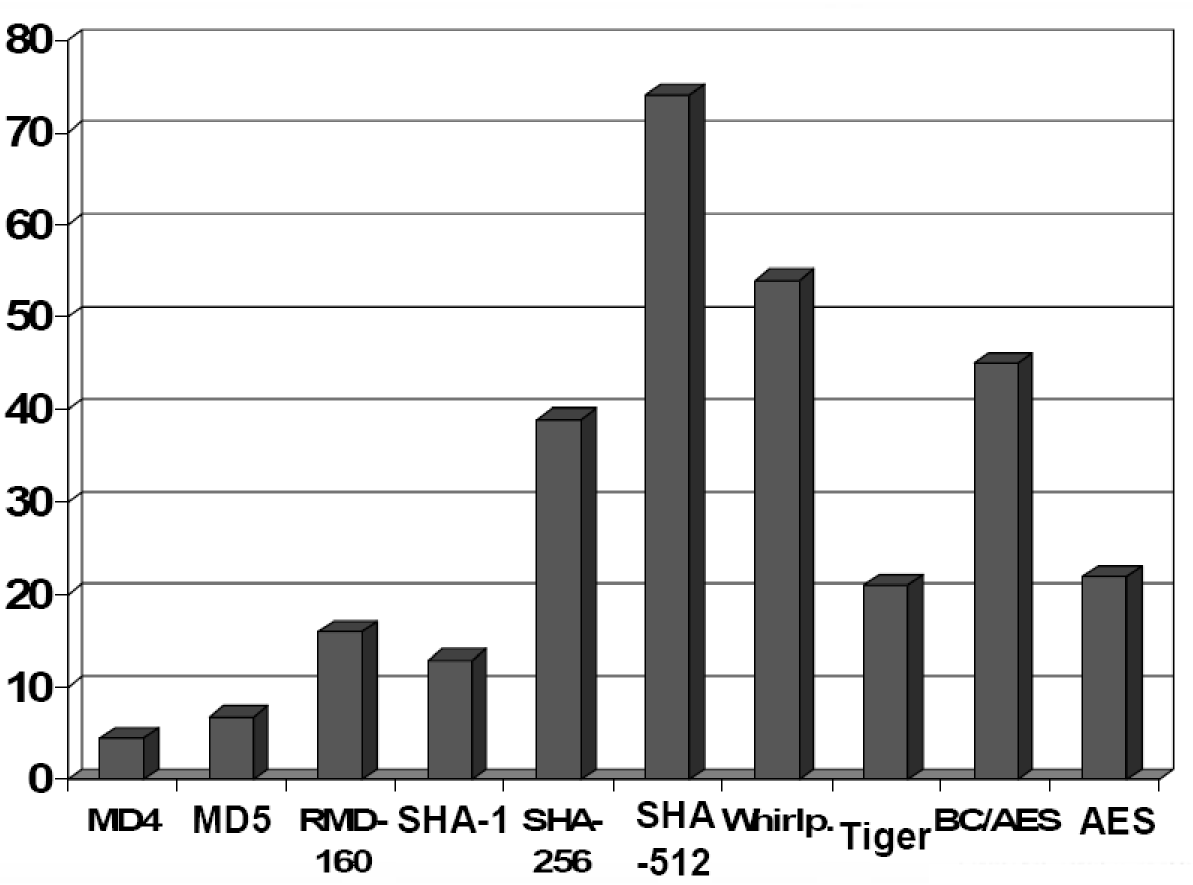
- Problem : **Design of high rate hash functions is everything but easy !**

Advanced Hash Standard

- MOTIVATION :
  - Many hash functions broken including standards – need for a new long-term standard
  - Variaty of designs not only MD iterated method

- PROBLEM :
  - We understand very little about hash functions
  - **New hash functions becomes slower then previous designs** !

Performance of hash functions



AHS timelines

- alternatives today:
  - RIPEMD-160 seems more secure than SHA-1 ☺
  - SHA-256, SHA-512
  - Whirlpool
- upgrading MD5 and SHA-1 in Internet protocols:
  - it doesn’t work: **algorithm flexibility is much harder than expected**
- randomized hashing

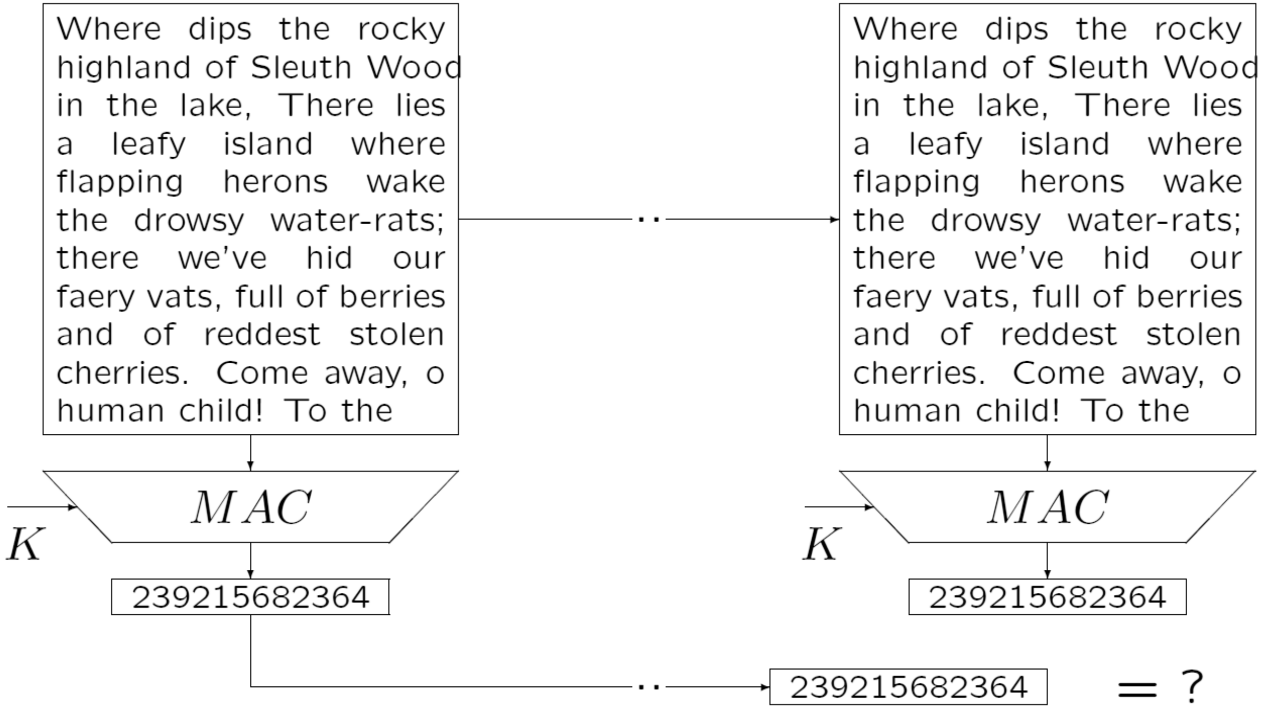
- NIST will run an open competition from 2008 to 2012

  The AHS must support 224, 256, 384, and 512-bit message digests, and must support a maximum message length of at least 2^{64} bits

  - 31 October 2008: submissions
  - February 2008: kickoff workshop
  - 2Q10 Announce finalists
  - 4Q11 Announce decision
  - 3Q12 Publish Advanced Hash Function Standard

Message Authentication Codes (MAC)

- Hash function with secret key



Where dips the rocky highland of Sleuth Wood in the lake, There lies a leafy island where flapping herons wake the drowsy water-rats; there we've hid our faery vats, full of berries and of reddest stolen cherries. Come away, o human child! To the

Where dips the rocky highland of Sleuth Wood in the lake, There lies a leafy island where flapping herons wake the drowsy water-rats; there we've hid our faery vats, full of berries and of reddest stolen cherries. Come away, o human child! To the

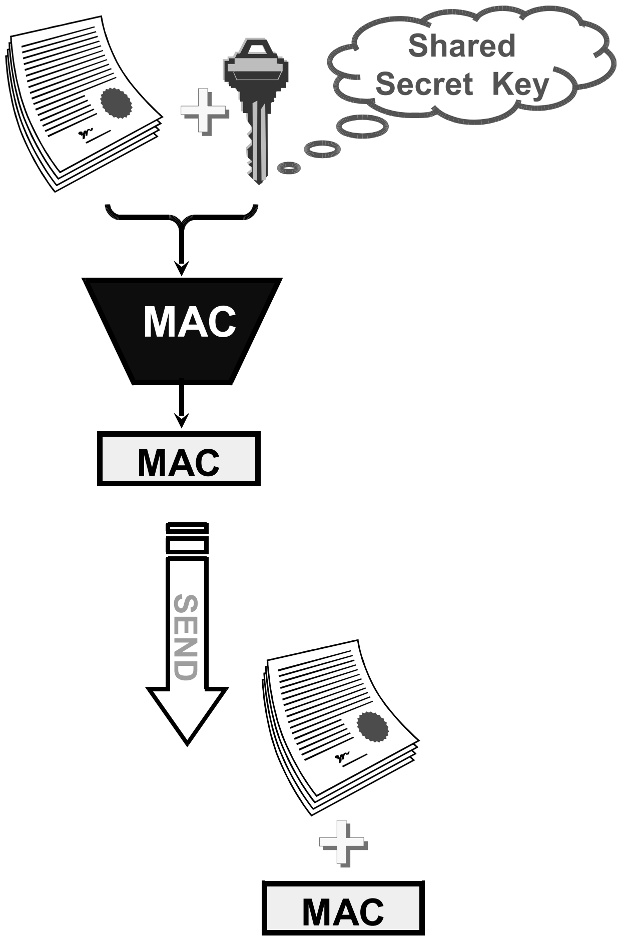
Why do we need MACs ?

- Hash function is public :
  - Provides message integrity
  - No message authentication (who created the message and message digest)
- But if secret key *K* is involved in algorithm you expect nobody else can create MAC but parties sharing the same key
- PROBLEM : Two parties share the key, who created the MAC then.
- No non-repudation property.

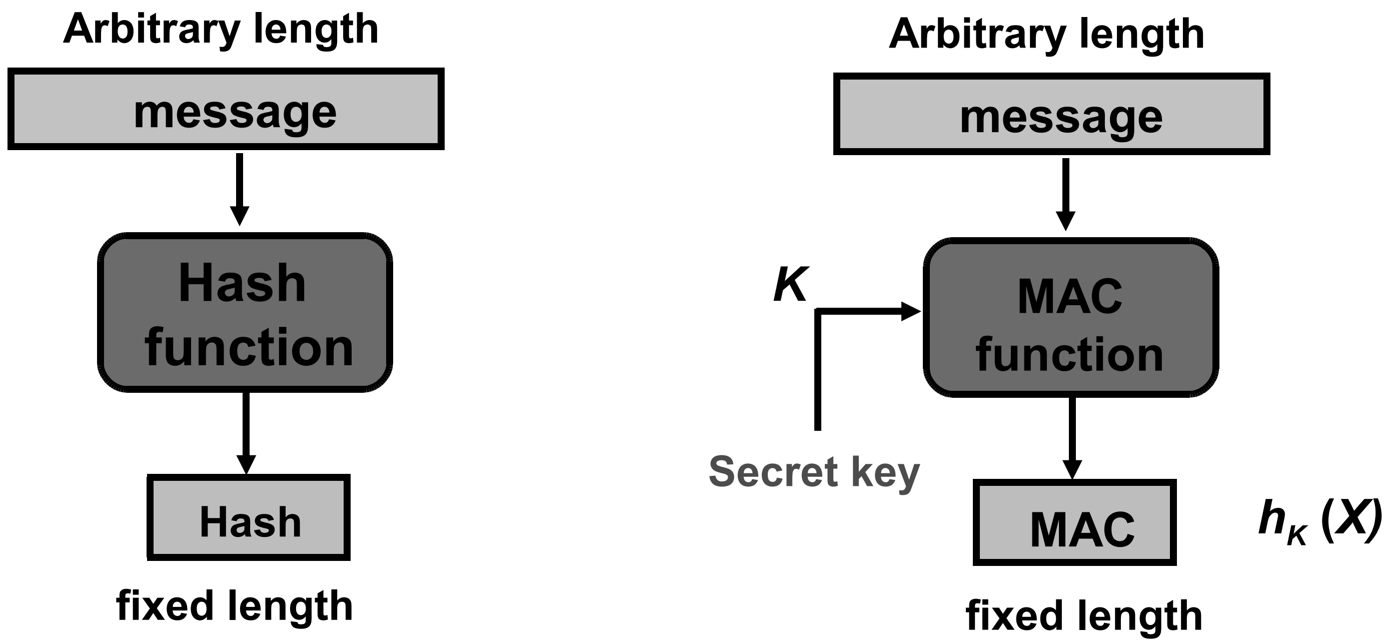
Message Authentication Codes (MACs)

- ***MAC***
  - ✓ Generate a fixed length MAC for an arbitrary length message
  - ✓ A keyed hash function
  - ✓ Message origin authentication
  - ✓ Message integrity
  - ✓ Entity authentication
  - ✓ Transaction authentication

- ***Constructions***
  - ✓ Keyed hash: HMAC, KMAC
  - ✓ Block cipher: CBC-MAC
  - ✓ Dedicated MAC: MAA, UMAC

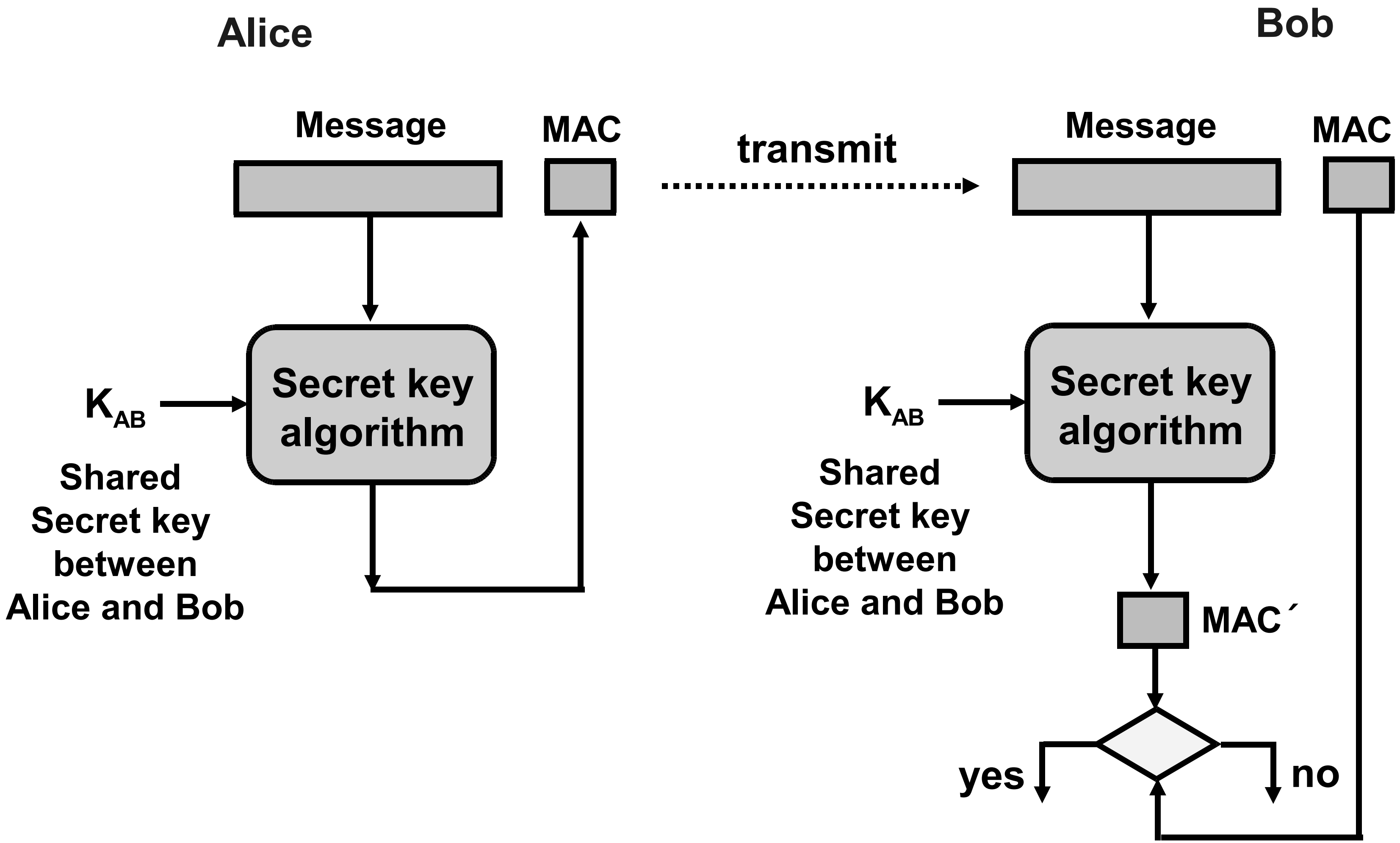


Comparison of Hash Function & MAC



- Easy to compute
- Compression: arbitrary length input to fixed length output
- Unkeyed function vs. Keyed function
- Computation of h_K (X) "hard" given only X even large number of pairs { X_i , h_K (X_i) } is available

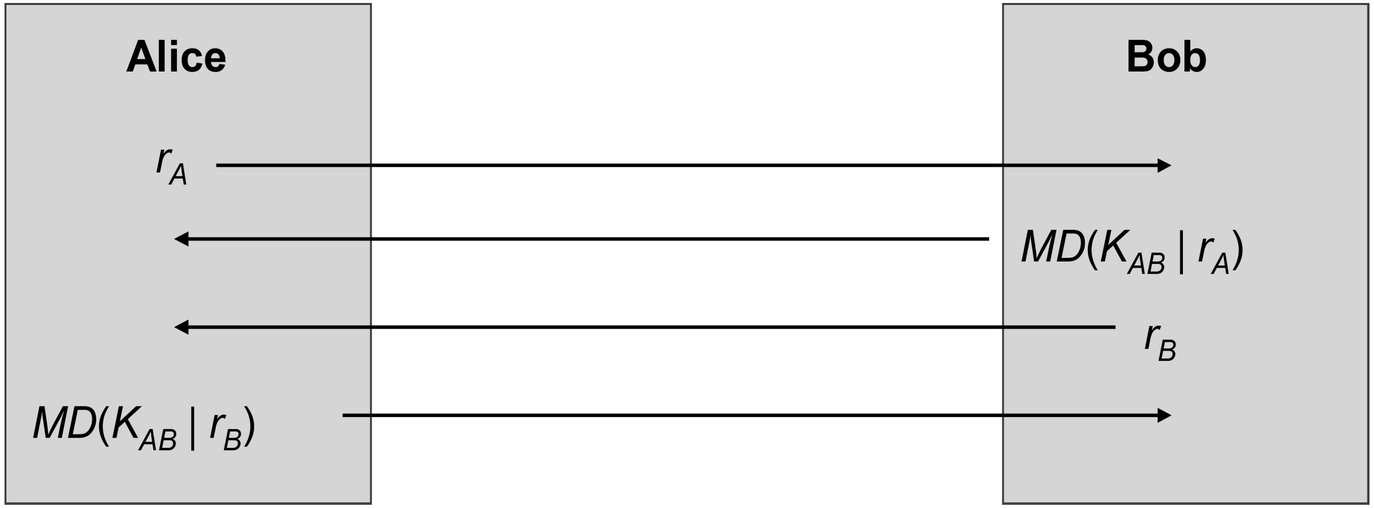
Message Authentication



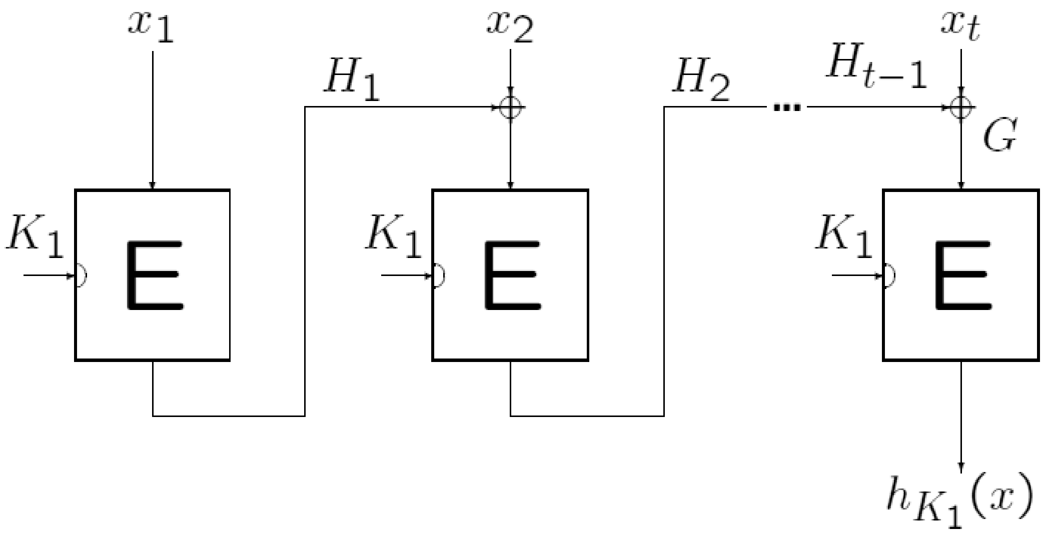
Authentication using keyed hash function

Authentication using a message digest:

- Alice and Bob share a **secret** K_{AB}
- Alice wants to know, if Bob is „still alive“: Alice sends a **challenge** r_A (a random number)
- Bob concatenates the secret K_{AB} with r_A and calculates a message digest $MD(K_{AB} \mid r_A)$
- Bob sends $MD(K_{AB} \mid r_A)$ to Alice, and Alice checks the result (apply the same procedure)
- Same procedure is applied in the other direction with a challenge r_B



CBC-MAC example

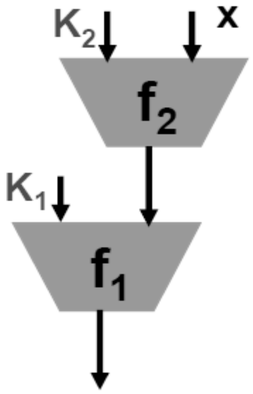


- ANSI 1982, FIPS 1985, ISO 1987,

HMAC scheme

HMAC based on MDx, SHA

- Widely used in SSL/TLS/IPsec
- Attacks not yet dramatic
- NMAC weaker than HMAC



	Rounds in f1	Rounds in f2	Data complexity
MD4	48	48	2^{88} CP & 2^{95} time
MD5	64	33 of 64	2^{126} CP
MD5	64	64	2^{51} CP & 2^{100} time (RK)
SHA(-0)	80	80	2^{109} CP
SHA-1	80	43 of 80	$2^{154.9}$ CP

End of overview

Course starts here

Definitions and taxonomy

- A **hash functions *h*** is a function that satisfies (as a minimum) :

  - **Compression –** h maps arbitrary input bitlength to a fixed output bitlength, say n, i.e. $h : \{0,1\}^*$ ---> $\{0,1\}^n$
  - **Ease of computation** – given h and an input x easy to compute $h(x)$

- **Additional desirable properties are:**

  - **Preimage resistance –** for all prespecified outputs it is computationally infeasible to find **any input** which hashes to the output, i.e. to find **any preimage** x' s.t. $h(x') = y$ when given any y for which corresponding input is not known.

2-nd preimage and precomputation of hash values

2. **2-nd preimage resistance** - computationally infeasible to find **any second input** that has the same hash value as **any specified input**, i.e. given x, to find x', x □ x' , s.t. $h(x) = h(x')$.

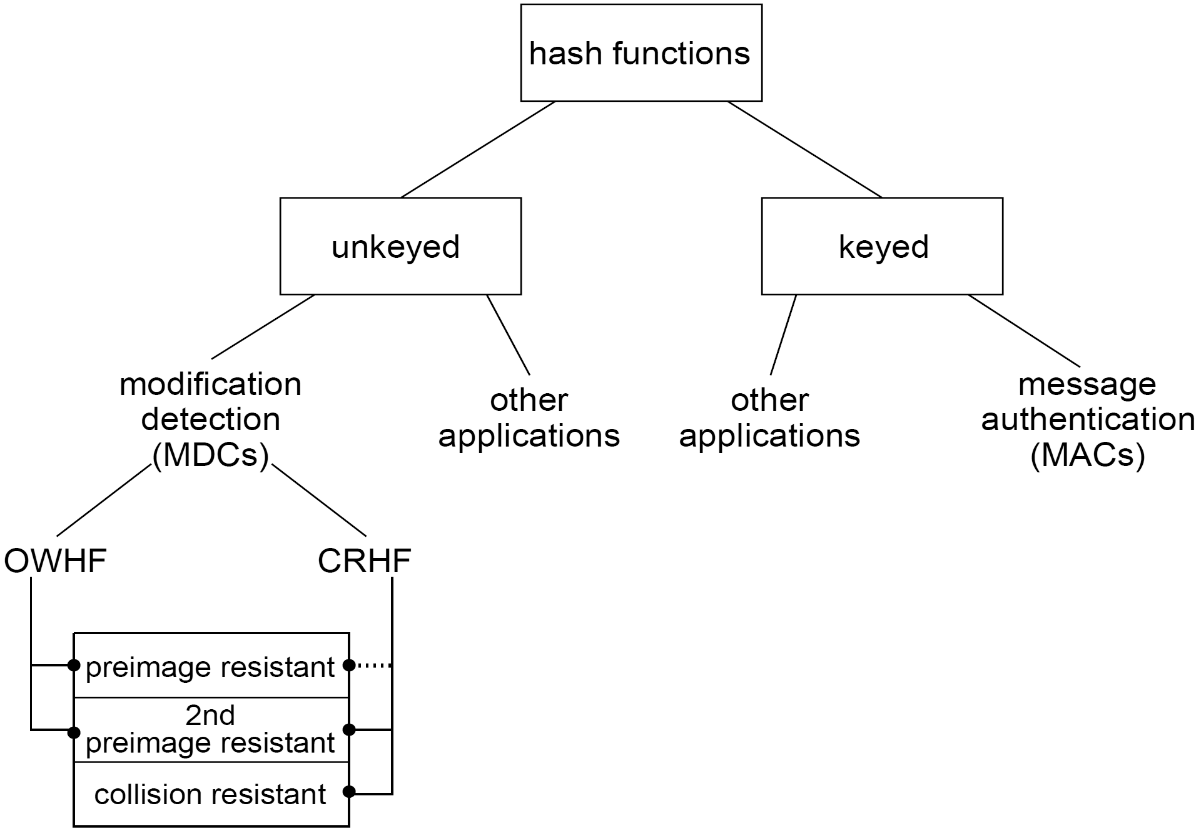
- Adversary may precompute outputs for a small number of inputs and invert hash function for these inputs. Time-memory attack, 64 bit hash:
  - Select 2^{40} random messages and compute hash values; store these in table
  - $O(2^{40})$ time and space for the precomputation
  - In active phase observe the hash value and compare with the table; probability of match (finding preimage) is $2^{40}/ 2^{64} = 2^{-24}$
  - Same reasoning is valid for the 2-nd preimage

Collision resistance, OWHF, CRHF

3. **Collision resistance –** it is computationally infeasible to find **any two distinct inputs** *x, x'* which hash to the same output, i.e. such that $h(x)=h(x')$

- Alternative terminology is :
  - Preimage resistant = one-way
  - 2-nd preimage resistant = weak collision resistant
  - Collision resistance = strong collision resistance
- **Definition:** A **one-way hash function** (**OWHF**) is a hash function which is preimage resistant and 2nd preimage resistant
- **Definition:** A **collision resistant hash function** (**CRHF**) is a hash function which is 2nd preimage resistant and collision resistant

Simplified classification



- In practice CRHF **almost** always includes preimage resistance

Relation between the properties

- **Theorem:** Collision resistance does not guarantee preimage resistance.

  Proof: Let *g* be a hash function which is collision resistant , *g* : $\{0,1\}^*$---> $\{0,1\}^n$. Consider h defined as,

$$h(x) = \begin{cases} 1 \| x, & \text{if } x \text{ has bitlength } n \\ 0 \| g(x), & \text{otherwise} \end{cases}$$

  Then h is $(n+1)$-bit hash function which is collision resistant but not preimage resistant (**details exercise**)

- The **example is more of pathological nature**, in practice collision resistance imply preimage resistance.

Relation between the properties II

- **Theorem:** Collision resistance implies 2-nd preimage resistance.

Proof: Suppose h is collision resistant. Fix an input x_j. If h is not 2-nd preimage resistant, then it is feasible to find some x_i such that $h(x_i)=h(x_j)$. This contradicts the assumption on collision resistance.

- **Fact:** Preimage resistance does not guarantee 2-nd preimage resistance.

Justification: $f(x) = x^2 \bmod n; n = pq,\ p,q$ large primes is one-way but second preimage is trivially $-x$.

Application - Example

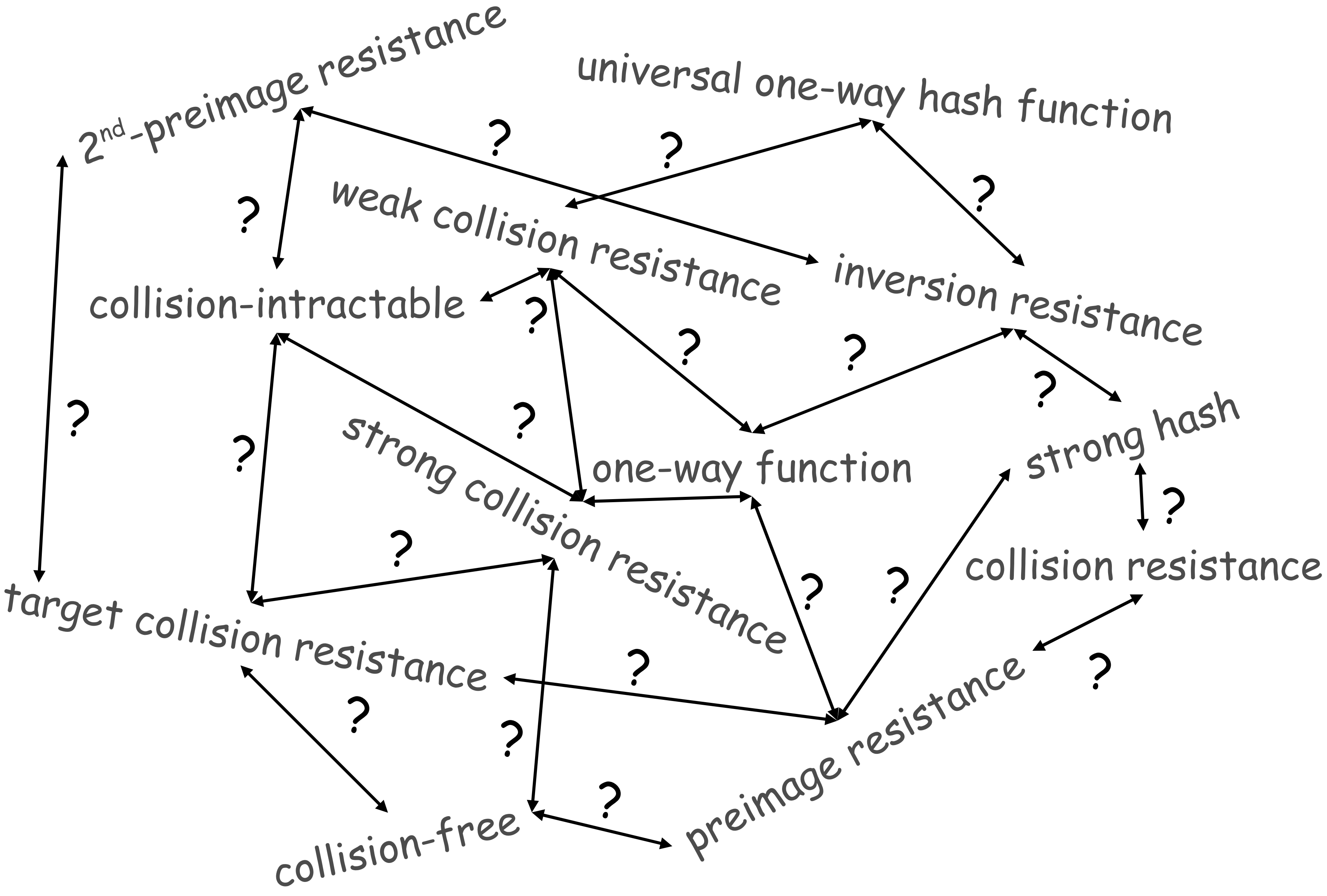
- DS (using RSA) is applied to hash value $h(x)$ rather than to message x.

- h should be 2nd preimage resistant for if not:
  - adversary *C* observes the signature of *A* on message x
  - *C* finds x' such that $h(x) = h(x')$
  - then *C* claims that *A* has signed x'
- If *C* can choose the message x that *A* signs then h should be CR:
  - *C* needs only to find collision pair (x, x')

- Preimage resistance is needed because :
  - *C* may take random y and compute $z = y^e \bmod n$ using public (e, n) and claim that y is *A*'s signature on z (**existential forgery)**
  - *A's* signature on z is $z^d = y^{ed} \bmod n = y$; Hence find x such that $h(x) = z$

Further concepts

- Apart from three standard security measures we have:
  - **Pseudo – preimage** (different IV's)
  - **Second pseudo-preimage** (different IV's)
  - **Collision for different IV's** (semi-free-start collision attack) – different IV's
  - **Pseudo- collision** (free-start collision attack) – free choice of IV's
  - **Non-correlation** (input and output bits not correlated)
  - **Near-collision resistance** ($h\ (x)$ and $h(x')$ differ in few bits)
  - **Partial-preimage resistance** (part of the input known still hard to recover the remainder)



2nd-preimage resistance

universal one-way hash function

weak collision resistance

inversion resistance

collision-intractable

one-way function

strong hash

strong collision resistance

collision resistance

target collision resistance

preimage resistance

collision-free