Cryptography

Lecture 7

Arpita Patra

Quick Recall and Today's Roadmap

>> AE: Two definitions (in one CCA-security was explicit and in the other it was implicit),

>> AE: Construction based on CPA secure SKE + CMA-secure MAC; proof of Security

>> Hash Function: Various Security Notions

>> Markle-Damgaard Domain Extension

>> Davis Meyer Hash function

>> Domain Extension for MAC using Hash function: Hash-and-Mac

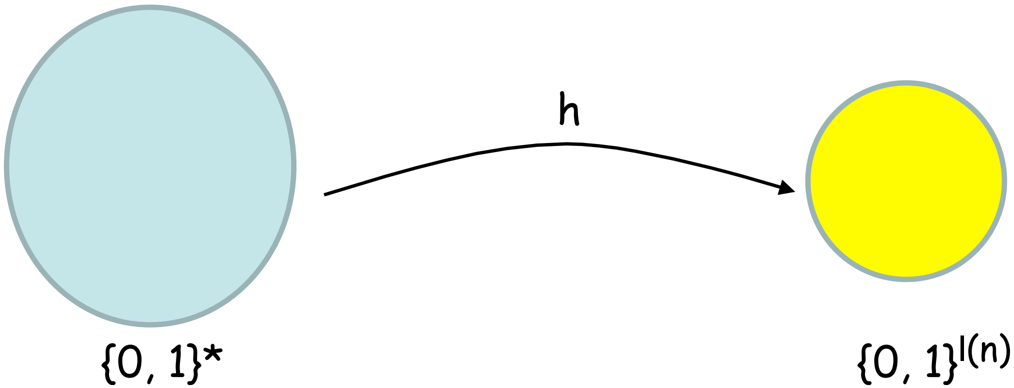
>> Key Agreement

>> Assumptions in Finite Cyclic groups - DL, CDH, DDH

- Groups
- Finite groups (modulo arithmetic)
- Finite cyclic groups
- Finite Cyclic groups of prime orders (special advantages)

Hash Functions

- Informally a hash-function is a (one-to-many) function mapping arbitrary-length bit-string to fixed-length bit-strings



- Usually |domain| >>>> |Co-domain| → collisions exist ($\exists\, x_1 \neq x_2$: $h(x_1) = h(x_2)$)

- Requirement from a good cryptographic hash function :

  - Given the description of h, finding collisions should be infeasible- Collision Resistance

  - Given the description of h, x and h(x) finding x′ with h(x′) = h(x) should be infeasible- Second Preimage Resistance

  - Given the description of h, given y = h(x) finding x′ with y = h(x′) should be infeasible- Preimage Resistance

Applications of Hash Functions

Application to MAC - Domain Extension) (hash) of file X

File X

- Message digest of a file serves as its unique identifier (unless a collision is found)
- The above idea has several applications
  - File Integrity Check
    - When a file is downloaded, its hash is also supplied, which is then compared with the hash of the downloaded file
  - Virus Fingerprinting
    - Virus scanners store the hashes of known viruses
    - When an email attachment or an application is downloaded, its hash is compared with the known hashes in the table to identify viruses
  - Deduplication
    - When a cloud storage is shared by several users, then storing the same file multiple times by multiple users is avoided by comparing the digests of uploaded files
  - Password Hashing

Hash Functions

Ivan Damgård:
Collision Free Hash Functions and Public Key Signature Schemes. EUROCRYPT 1987: 203-216

Collision Resistance Security

Experiment Hash-CR$_{A,\Pi}(n)$ $\qquad$ Π = (Gen, h), n

Attacker A

k

(x,x′)

Collision

I can break Π

Run time: Poly(n)

Let me verify

k

Gen(1^n)

game output

- 1 (A succeeds) if h(x) = h(x′)
- 0 (A fails) otherwise

Π is Collision Resistant HF if for every A, there is a negl(n) such that

$$\Pr[\text{Hash-CR}_{A,\Pi}(n) = 1] \leq \text{negl}(n)$$

Second Preimage Resistance Security

Experiment $\text{Hash-SPR}_{A,\Pi}(n)$ Π = (Gen, h), n

Attacker A

I can break Π

Run time: Poly(n)

k and a uniform x

x′

Let me verify

k

Gen(1^n)

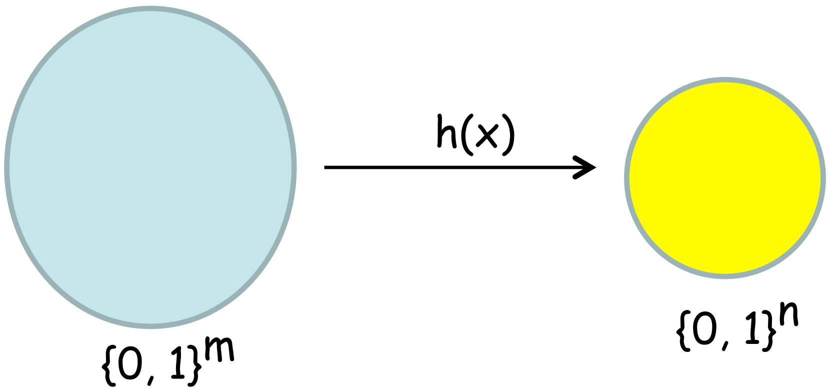
game output

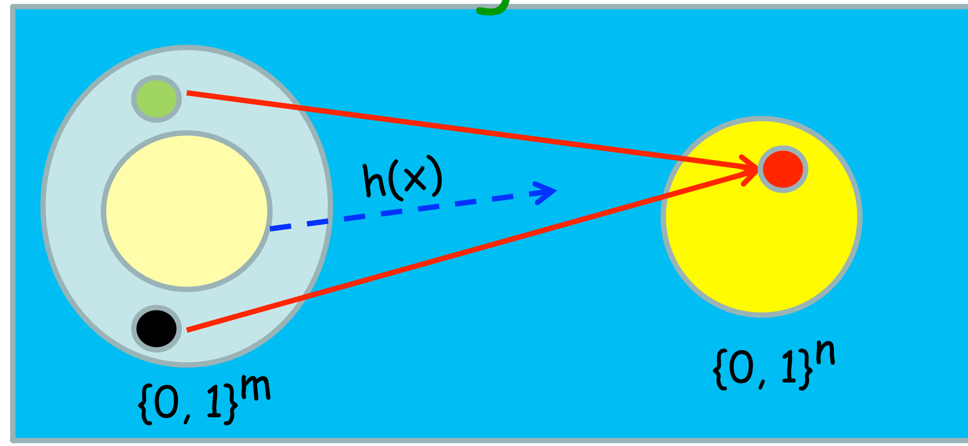
- 1 (A succeeds) if h(x) = h(x′)
- 0 (A fails) otherwise

Π is second preimage resistant HF if for every A, there is a negl(n) such that

$$\Pr[\text{Hash-SPR}_{A,\Pi}(n) = 1] \leq \text{negl}(n)$$

Collision Resistance & Second Preimage Resistance





g(x)

- Collision Resistance → second preimage resistance. Otherway?
- Let h: $\{0, 1\}^m \rightarrow \{0, 1\}^n$ be a second preimage resistant hash function
- We can design a new hash function from h which is second preimage resistant but not collision resistant ?
- Define a new hash function g: $\{0, 1\}^m \rightarrow \{0, 1\}^n$ as follows:

  - $$g(x) = \begin{cases} 0^n, \text{ if } x = 0^m \text{ or } x = 1^m \\ h(x), \text{ otherwise} \end{cases}$$

- g is collision resistant with probability 0
- If h is second preimage resistant with probability negl() then g is second preimage resistant with probability = $1/2^{m-1}$ + negl() = negligible

Preimage Resistance Security

Experiment Hash-PR$_{A,\Pi}(n)$

Π = (Gen, h), n

Attacker A

I can break Π

Run time: Poly(n)

k and uniform y

x

Let me verify

k

Gen(1^n)

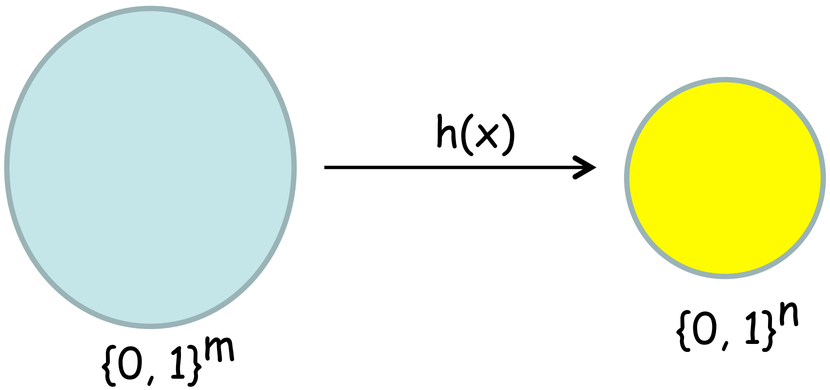
game output

- 1 (A succeeds) if h(x) = y
- 0 (A fails) otherwise

Π Is Preimage Resistant HF if for every A, there is a negl(n) such that

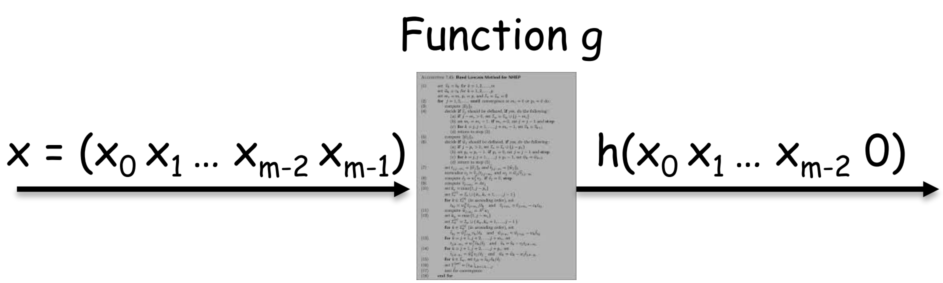
$$\Pr[\text{Hash-PR}_{A,\Pi}(n) = 1] \leq \text{negl}(n)$$

Pre-image Resistance $\nrightarrow$ Second Pre-image Resistance



- Let $h: \{0, 1\}^m \rightarrow \{0, 1\}^n$ be a pre-image resistant hash function

- Define a new hash function $g: \{0, 1\}^m \rightarrow \{0, 1\}^n$ as follows:



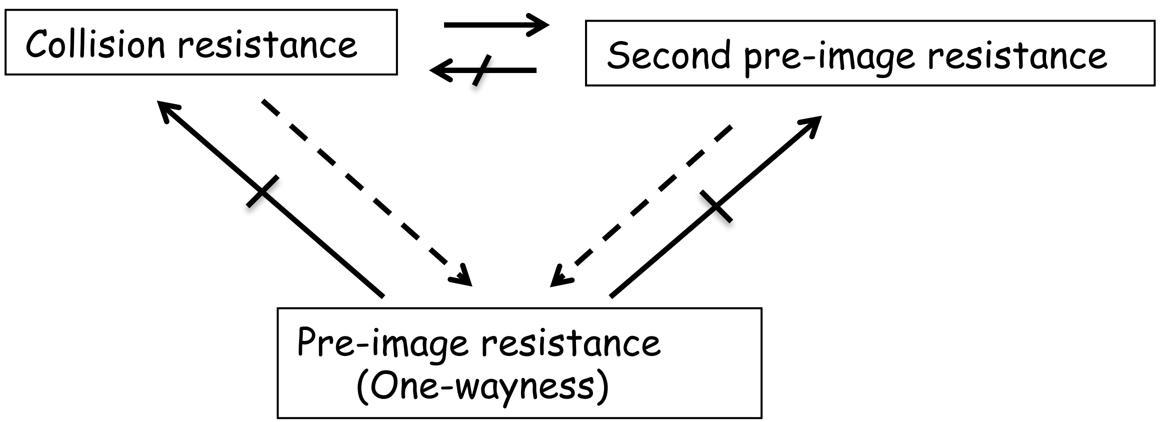
- If h is pre-image resistant with probability negl() then g is pre-image resistant with probability at least 2 negl() = negligible

- g is second-preimage resistant with probability 0

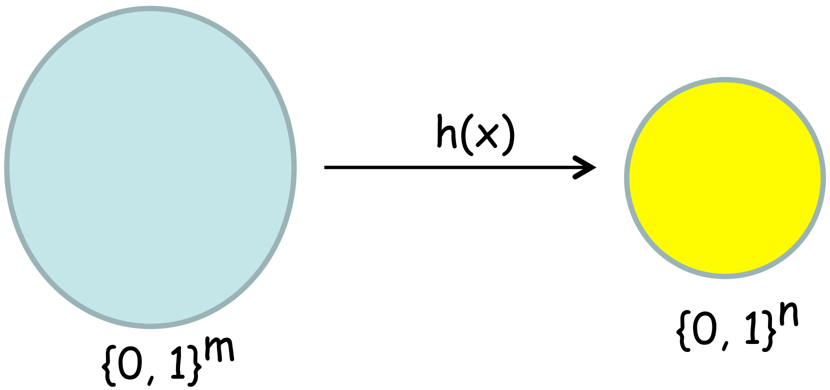
  - Given a random x and g(x), trivial to find $x' \neq x$ with $g(x') = g(x)$

    - x' is the whole x with final bit flipped --- in fact g is also not collision-resistant

Relation among Security Notions

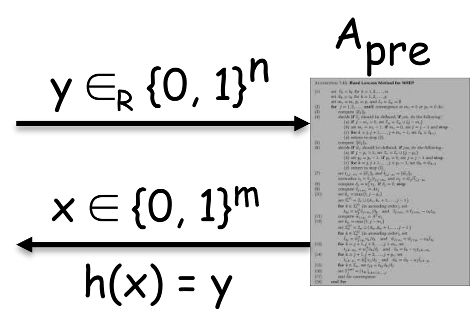


Second Preimage Resistance and Preimage Resistance

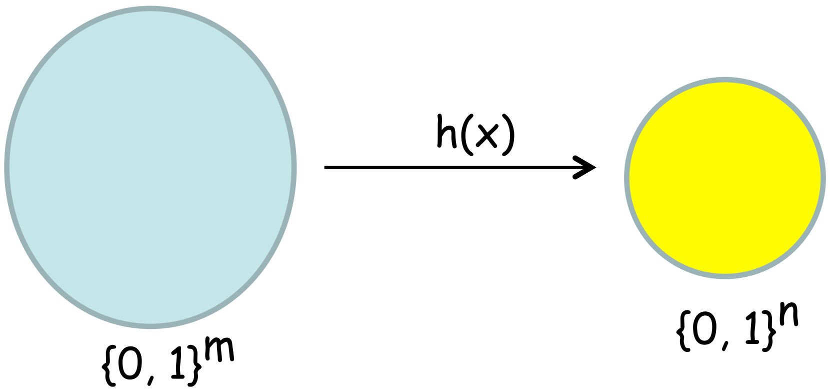


- Let $h: \{0, 1\}^m \rightarrow \{0, 1\}^n$ be a second-preimage resistant hash function
  - Does it imply that h is also pre-image resistant ?
  - Depends upon the compression ratio !!

- Suppose h is not pre-image resistant --- PPT algorithm A_{pre} for computing pre-image
  - Then consider the following PPT algorithm A_{sec} for computing second pre-images corresponding to random x and h(x)

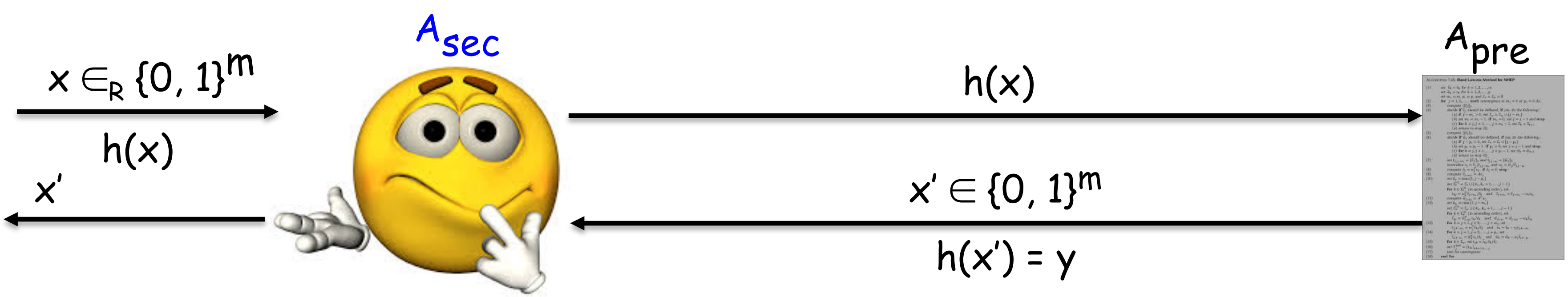


Second Preimage Resistance and Preimage Resistance



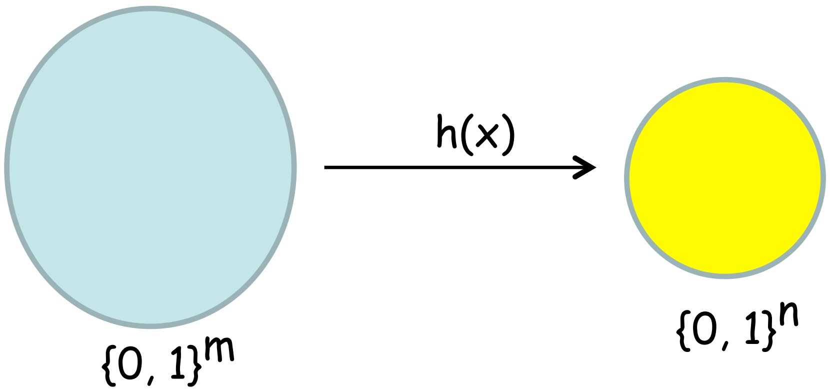
- Let $h: \{0, 1\}^m \rightarrow \{0, 1\}^n$ be a second-preimage resistant hash function
  - Does it imply that h is also pre-image resistant ?
  - Depends upon the compression ratio !!

- Suppose h is not pre-image resistant --- PPT algorithm A_{pre} for computing pre-image
  - Then consider the following PPT algorithm A_{sec} for computing second pre-images corresponding to random x and h(x)



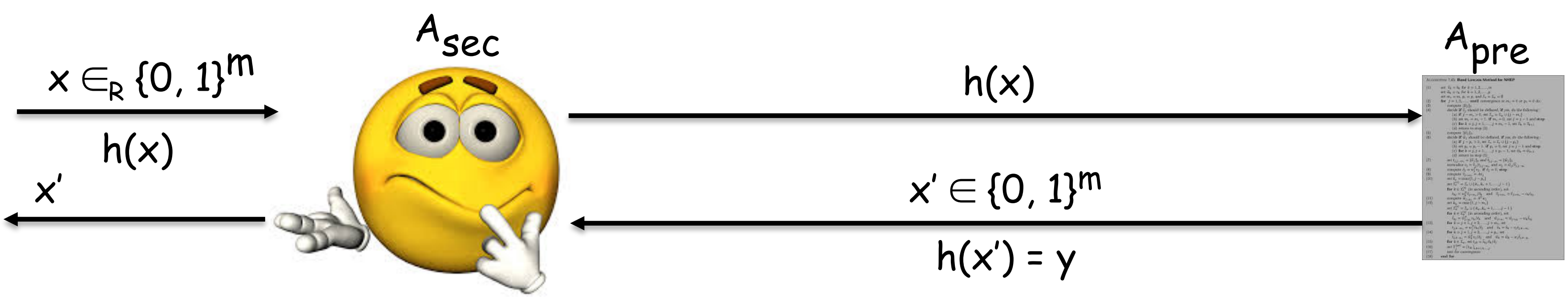
- What is the probability that A_{sec} outputs $x' \neq x$? --- depends upon compression ratio
  - Ex: if $m = 2n$, then on an average every two different x values mapped to the same y. So with probability roughly $1\text{-}2^{-n}$, $x' \neq x \rightarrow$ h is not second-preimage resistant (contradiction)

Second Preimage Resistance and Preimage Resistance



- Let $h: \{0, 1\}^m \rightarrow \{0, 1\}^n$ be a second-preimage resistant hash function
  - Does it imply that h is also pre-image resistant ?
  - Depends upon the compression ratio !!

- Suppose h is not pre-image resistant --- PPT algorithm A_{pre} for computing pre-image
  - Then consider the following PPT algorithm A_{sec} for computing second pre-images corresponding to random x and h(x)



- What is the probability that Asec outputs $x' \neq x$? --- depends upon compression ratio
  - Ex: if m = n (say the identity function), then $x' \neq x$ with probability 0 $\nrightarrow$ h is not second-preimage resistant (no contradiction)

Constructing Hash Functions

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

>> Stage I: $h: \{0, 1\}^{l'(n)} \rightarrow \{0, 1\}^{l(n}$

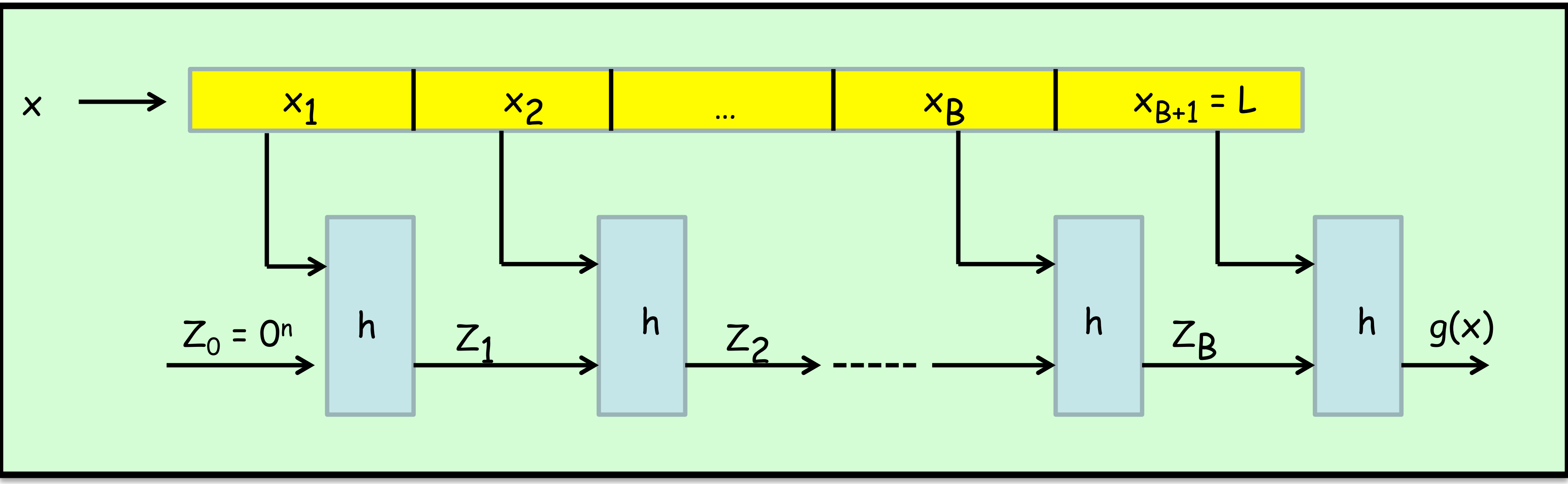
Implies compressing by bit as hard (easy) as compressing arbitrary number of bits

>> Stage II: Domain Extension

The Merkle-Damgaard Transform

- Given: A fixed-length collision-resistant function $h: \{0,1\}^{2n} \rightarrow \{0,1\}^n$
- Goal: A arbitrary-length collision-resistant function $h: \{0,1\}^* \rightarrow \{0,1\}^n$ $* < 2^n$

Divide input x into blocks of length n --- $B = L/n$ (use 0-padding to make L a multiple of n)



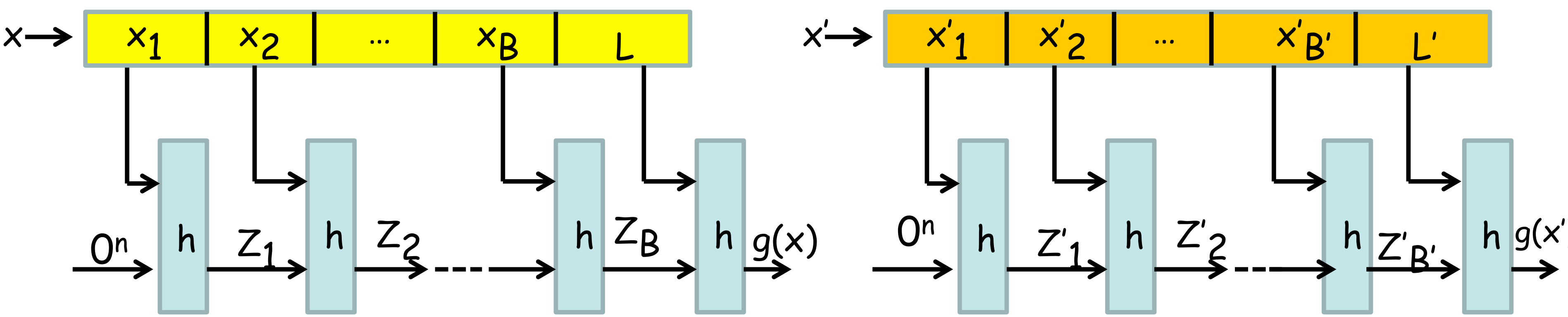
Used Everywhere in practice! SHA2, MD5

The Merkle-Damgard Transform: Security

Theorem: If h is a hash function for messages of length 2n, then the Merkle-Damgard transformation yields a collision-resistant hash function for arbitrary length messages.

Proof: Reduction yet again!

- If Merkle-Damgard is not collision-resistant then h is also not collision resistant
- Let $x = (x_1\ x_2 \ldots x_B\ L)$ and $x' = (x'_1\ x'_2 \ldots x'_{B'}\ L')$ be two different messages of length L and L' respectively, such that $g(x) = g(x')$
- Case I: $L' \neq L$:

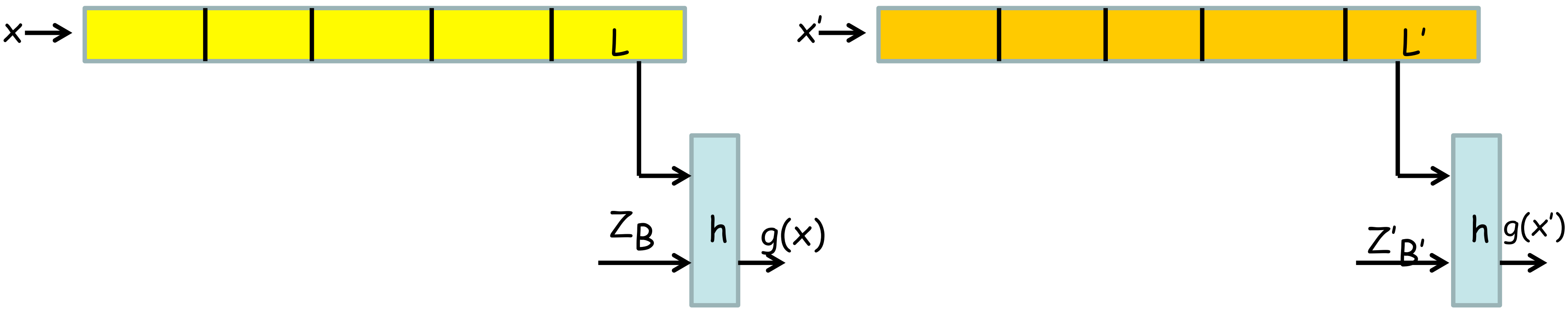


- Can you spot a collision for h in this case ?

The Merkle-Damgard Transform: Security

Theorem: If h is a hash function for messages of length 2n, then the Merkle-Damgard transformation yields a collision-resistant hash function for arbitrary length messages.

- If Merkle-Damgard is not collision-resistant then h is also not collision resistant
- Let $x = (x_1\ x_2\ \ldots\ x_B\ L)$ and $x' = (x'_1\ x'_2\ \ldots\ x'_{B'}\ L')$ be two different messages of length L and L′ respectively, such that $g(x) = g(x')$
- Case I: $L' \neq L$:

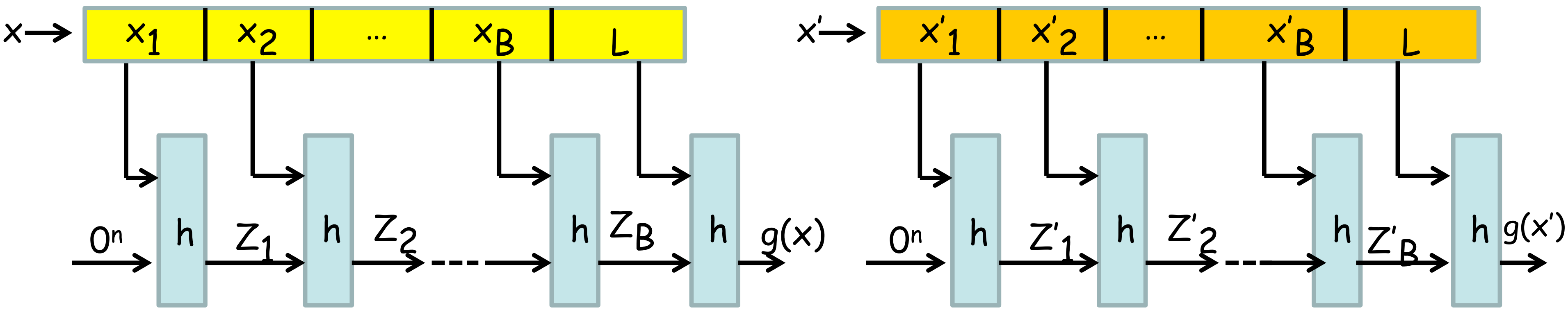


- Can you spot a collision for h in this case ?
  - $(Z_B \,||\, L) \neq (Z'_{B'} \,||\, L')$ is a collision for h --- contradiction

The Merkle-Damgard Transform: Security

Theorem: If h is a hash function for messages of length 2n, then the Merkle-Damgard transformation yields a collision-resistant hash function for arbitrary length messages.

- If Merkle-Damgard is not collision-resistant then h is also not collision resistant
- Let $x = (x_1\ x_2\ \ldots\ x_B\ L)$ and $x' = (x'_1\ x'_2\ \ldots\ x'_{B'}\ L')$ be two different messages of length L and L' respectively, such that $g(x) = g(x')$
- Case II: $L' = L$:

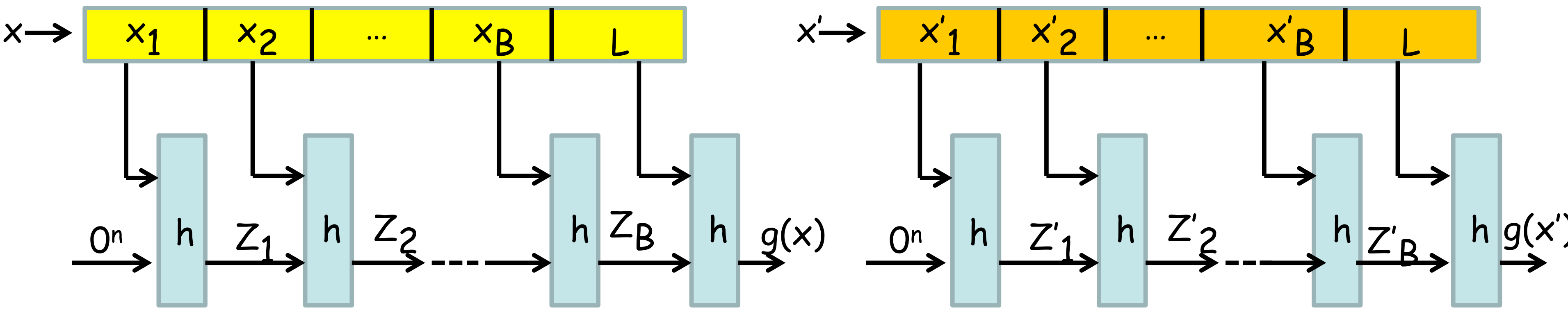


- Can you spot a collision for h in this case ?

The Merkle-Damgard Transform: Security

Theorem: If h is a hash function for messages of length 2n, then the Merkle-Damgard transformation yields a collision-resistant hash function for arbitrary length messages.

- If Merkle-Damgard is not collision-resistant then h is also not collision resistant
- Let $x = (x_1\ x_2 \ldots x_B\ L)$ and $x' = (x'_1\ x'_2 \ldots x'_{B'}\ L')$ be two different messages of length L and L' respectively, such that $g(x) = g(x')$
- Case II: $L' = L$:

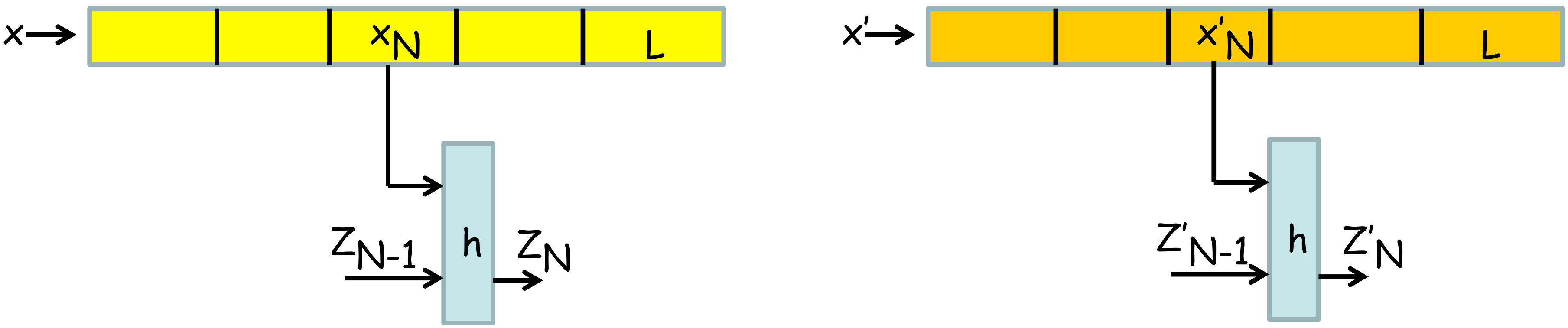


- Can you spot a collision for h in this case ?
  - Define $I_i = (x_i \mid\mid Z_{i-1})$ and $I'_i = (x'_i \mid\mid Z'_{i-1})$ --- inputs for the i^{th} invocation of h
  - Let N be the largest index with $I_N \neq I'_N$ --- such an N always exist

The Merkle-Damgard Transform: Security

Theorem: If h is a hash function for messages of length 2n, then the Merkle-Damgard transformation yields a collision-resistant hash function for arbitrary length messages.

- If Merkle-Damgard is not collision-resistant then h is also not collision resistant
- Let $x = (x_1\ x_2 \ldots x_B\ L)$ and $x' = (x'_1\ x'_2 \ldots x'_{B'}\ L')$ be two different messages of length L and L' respectively, such that $g(x) = g(x')$
- Case II: $L' = L$:

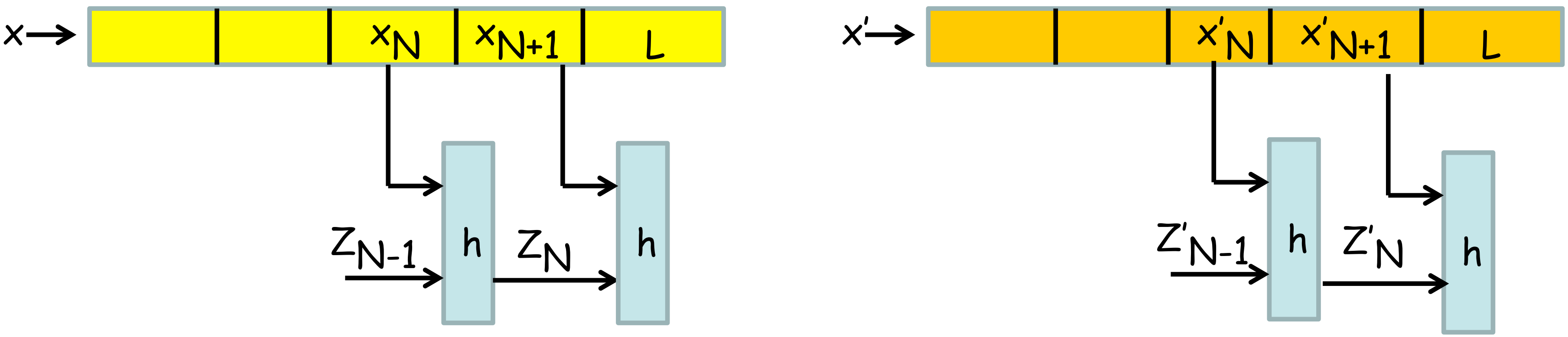


- By maximality of N, $Z_N = Z'_N$ as $I_{N+1} = I'_{N+1}$ and so on

The Merkle-Damgard Transform: Security

Theorem: If h is a hash function for messages of length 2n, then the Merkle-Damgard transformation yields a collision-resistant hash function for arbitrary length messages.

- If Merkle-Damgard is not collision-resistant then h is also not collision resistant
- Let $x = (x_1\ x_2\ \ldots\ x_B\ L)$ and $x' = (x'_1\ x'_2\ \ldots\ x'_{B'}\ L')$ be two different messages of length L and L' respectively, such that $g(x) = g(x')$
- Case II: $L' = L$:



- By maximality of N, $Z_N = Z'_N$ as $I_{N+1} = I'_{N+1}$ and so on
- So $h(I_N) = h(I'_N)$, even though $I_N \neq I'_N$
  - (I_N, I'_N) constitutes a collision for h --- a contradiction

Constructing Hash Functions

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

>> Stage I: $h: \{0, 1\}^{l'(n)} \rightarrow \{0, 1\}^{l(n)}$; $l'(n) > l(n)$

>> Stage II: Domain Extension

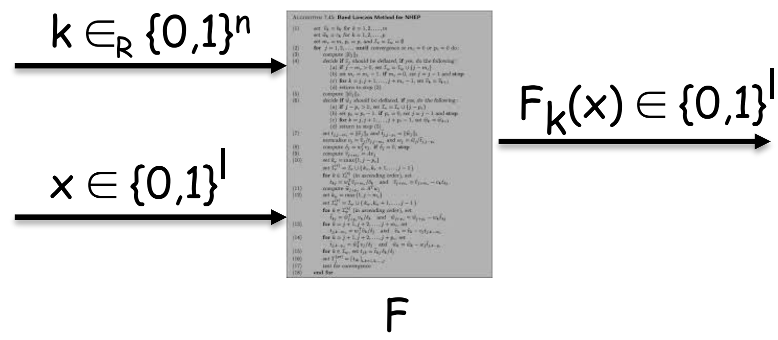
>> Davies-Meyer construction,

>> Matyas-Meyer-Oseas construction,

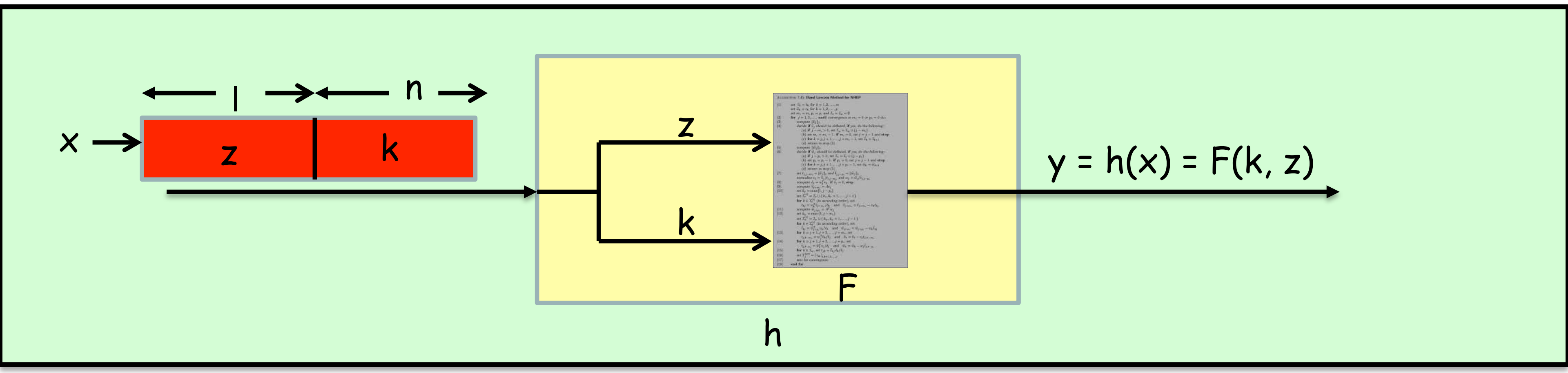
>> Miyaguchi-Preneel construction, etc

>> Heuristics.
>> None of them are provably secure
>> Weak guarantees of them being collision resistant is known☹

Davis-Meyer Construction

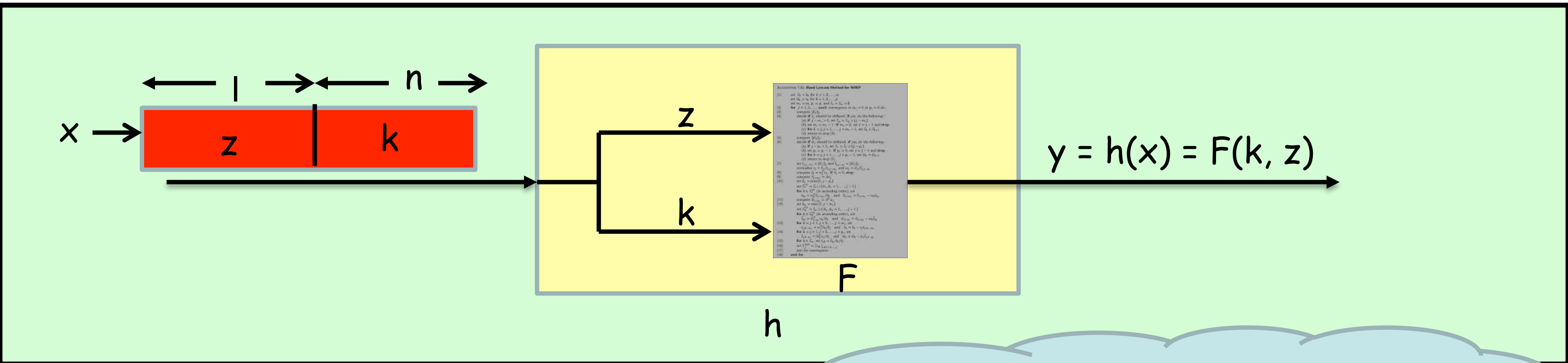


- Given :
  - A SPRP $F: \{0, 1\}^n \times \{0, 1\}^l \rightarrow \{0, 1\}^l$
- Goal :
  - A fixed-length hash function $h: \{0, 1\}^{l+n} \rightarrow \{0, 1\}^l$



- Is h a collision-resistant compression function ?

Davis-Meyer Construction

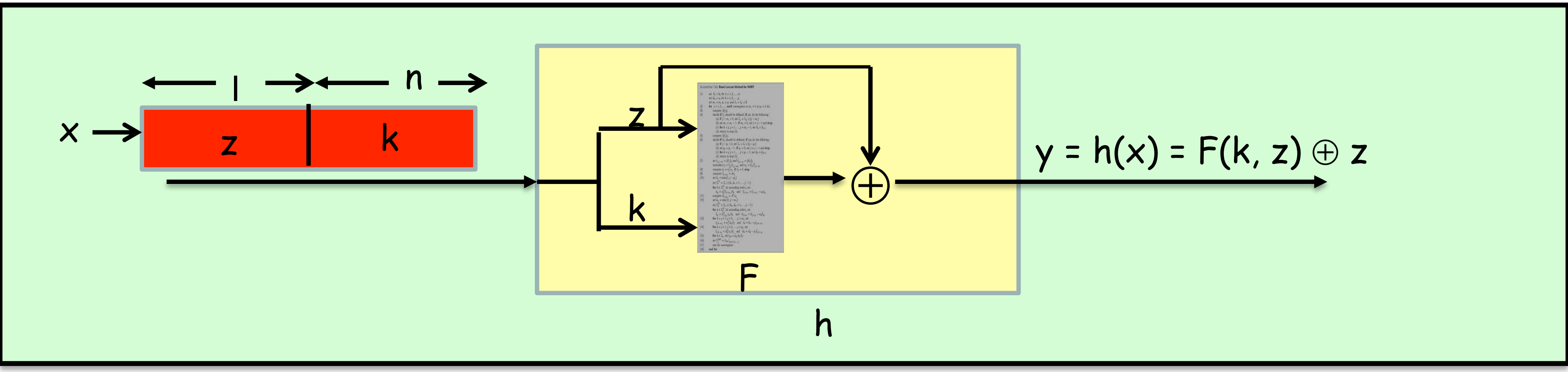


How to prevent such attack?

Easy to find collision assuming F to be SPRP.

$x = z||k$ ⇒ $y = F(k,z)$ ⇒ $z' = F^{-1}(k', F(k,z))$ ⇒ $x' = z' \,||\, k'$

Davis-Meyer Construction



5th Chalk and Talk topic

Part I: Proof of the theorem below

Part II: Birthday Attack OR Time/Space Tradeoff for Inverting Functions

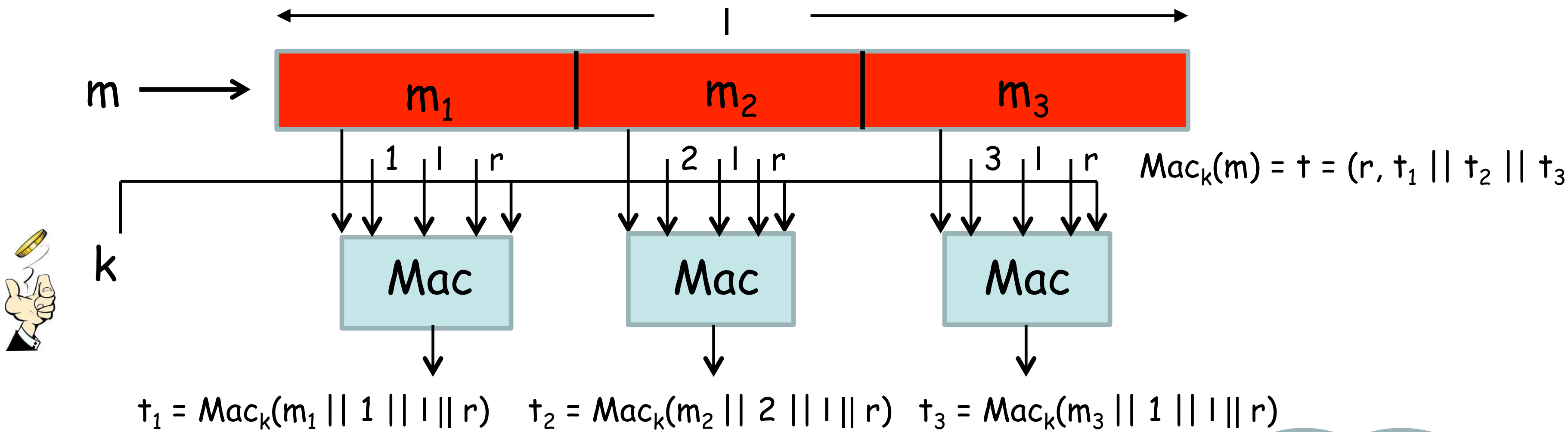
Theorem: If F is a ideal random strong permutation, then adversary making $q < 2^{l/2}$ queries finds a collision with probability $q^2/2^l$

Practical Construction of Hash Functions

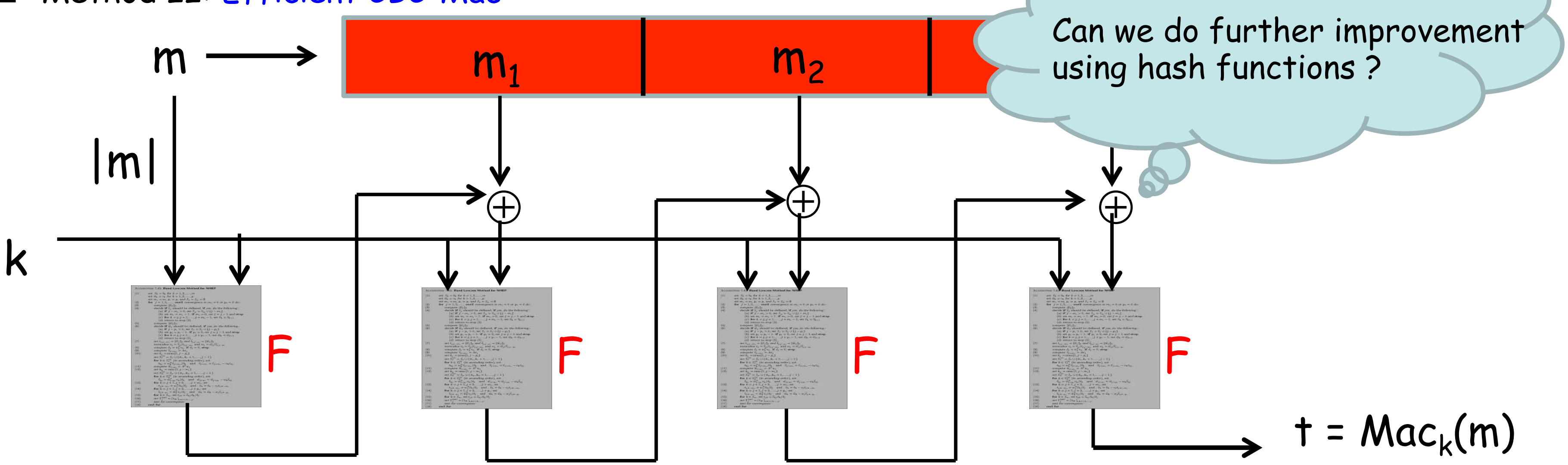
- MD5 :
  - 128-bit output; designed in 1991 and believed to be secure (collision-resistant)
  - Completely broken in 2004 by Chinese cryptanalysts; collision can be found in less than a minute on a desktop PC
- SHA (Secure Hash Algorithm) Family
  - Standardized by NIST. Got two flavors SHA-1 and SHA-2
  - First a fixed-length compression function designed from a block cipher
  - In the second stage, the Merkle-Damgard transformation is applied
  - Special block ciphers designed for the stage I
- SHA-3 (Keccak)
  - Winner of the NIST competition for hash functions
  - Construction very different from previous constructions
  - For stage I uses an un-keyed permutation of block length 1600 bits
  - For stage II uses a new approach called sponge construction

Message Authentication Using Hash Functions

- Given a fixed-length MAC, we can design arbitrary-length MAC using two methods:
- Method I: Generic (randomized) but inefficient construction

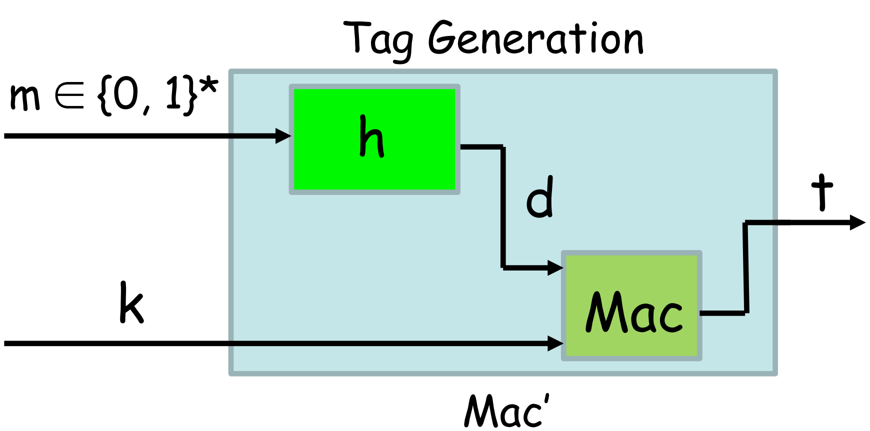


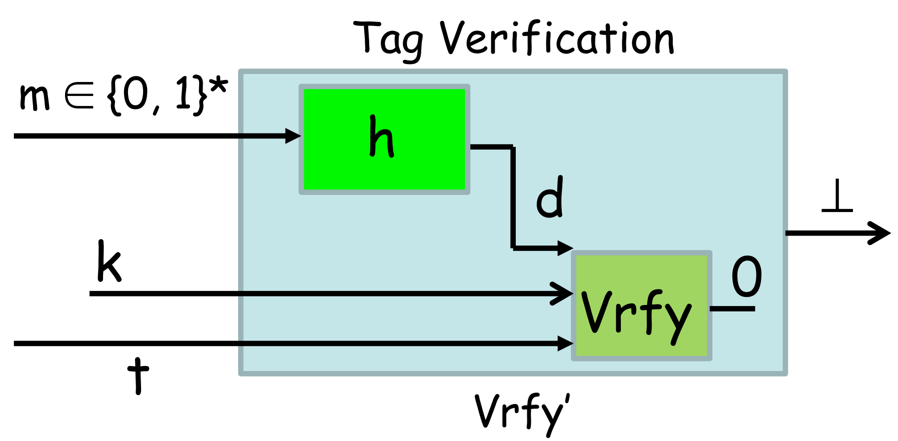
- Method II: Efficient CBC-Mac



Message Authentication Using Hash Functions (Hash-and-MAC Paradigm)

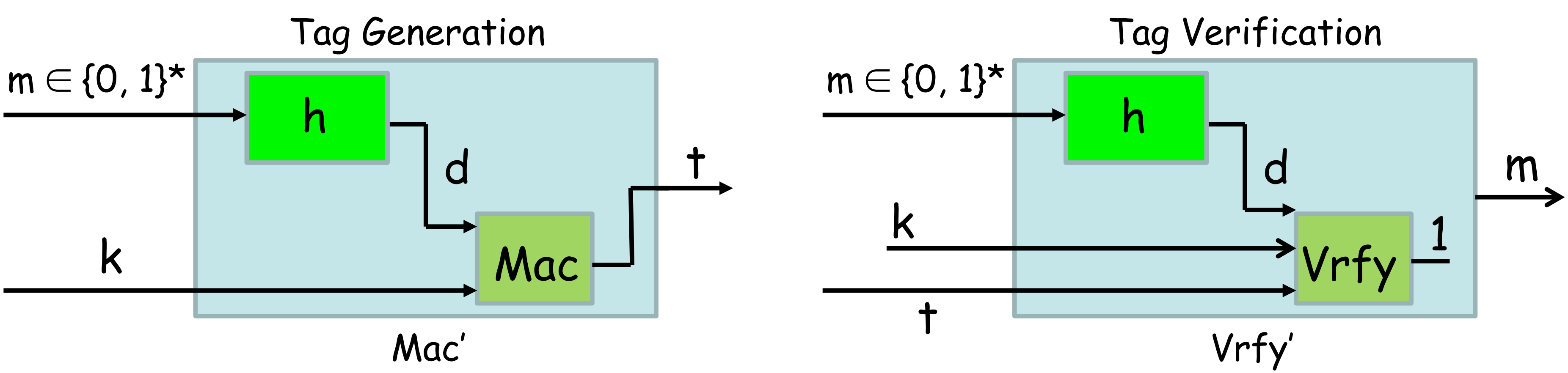
- Given an arbitrary-length message, compute its Mac-tag in two stages:
  - Step I: Compress the arbitrary-length message to a fixed-length string using a CRHF
  - Step II: Compute the Mac-tag on the message digest (output of the CRHF)
- Let:
  - Π_{MAC} = (Mac, Vrfy) be a MAC for messages of length $l(n)$
  - h: $\{0, 1\}^* \rightarrow \{0, 1\}^{l(n)}$ be a collision-resistant hash function
- Then Π'_{MAC} = (Mac', Vrfy') is a MAC for arbitrary-length messages constructed as follows:





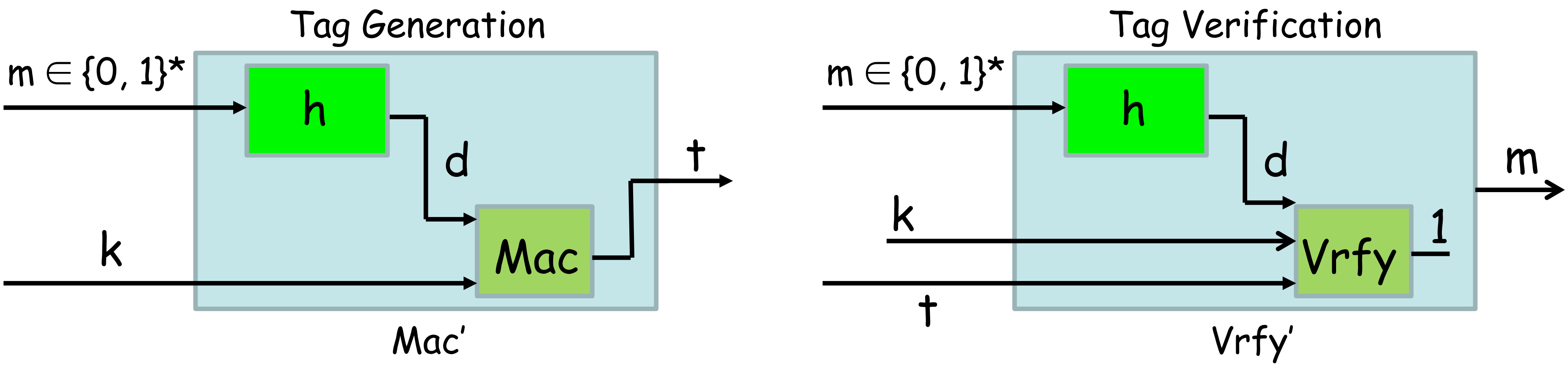
Message Authentication Using Hash Functions (Hash-and-MAC Paradigm)

- Given an arbitrary-length message, compute its Mac-tag in two stages:
  - Step I: Compress the arbitrary-length message to a fixed-length string using a CRHF
  - Step II: Compute the Mac-tag on the message digest (output of the CRHF)
- Let:
  - Π_{MAC} = (Mac, Vrfy) be a MAC for messages of length $l(n)$
  - h: $\{0, 1\}^* \rightarrow \{0, 1\}^{l(n)}$ be a collision-resistant hash function
- Then Π'_{MAC} = (Mac', Vrfy') is a MAC for arbitrary-length messages constructed as follows:

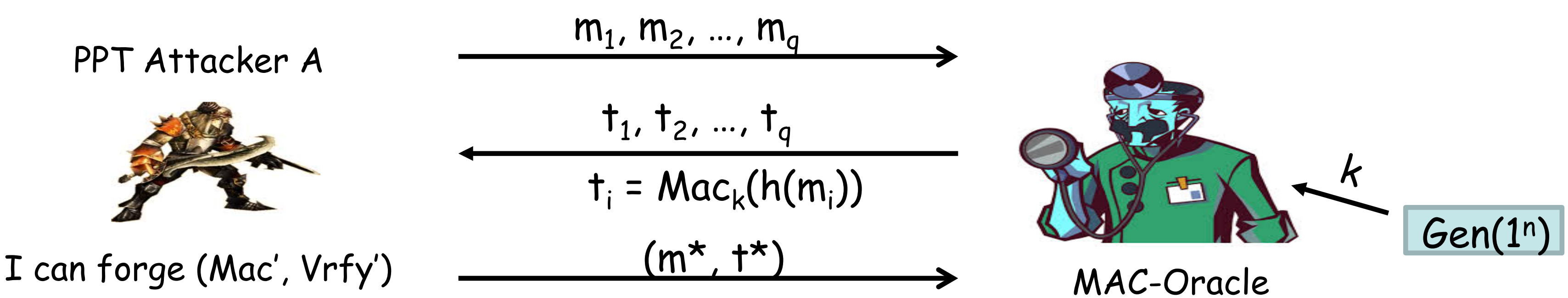


- The above construction is more efficient than CBC-Mac --- is it secure ?

Hash-and-MAC Paradigm: Security (Sketch)

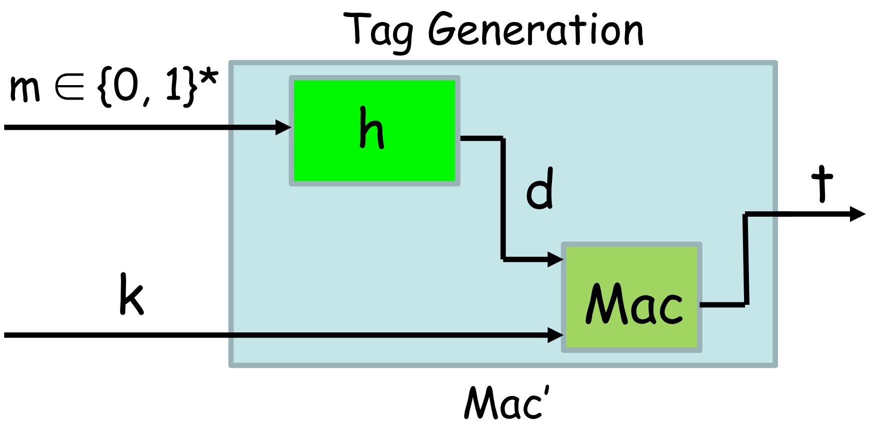


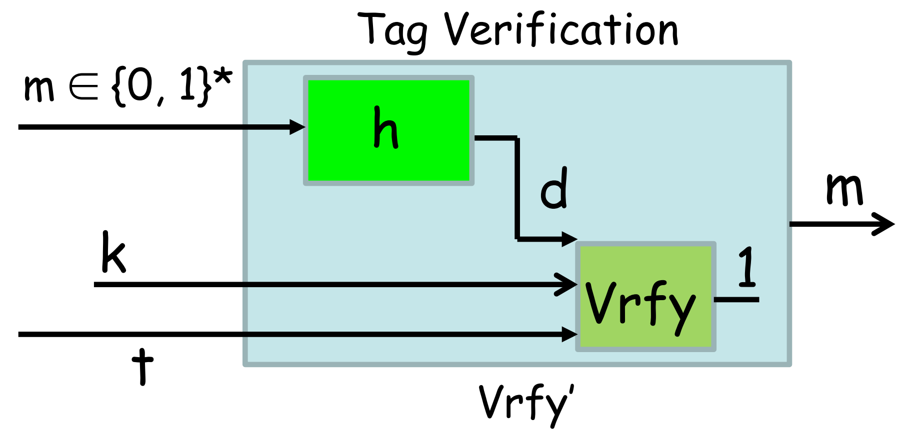
- The above construction gives a secure MAC for arbitrary-length messages



- A successfully forges (Mac', Vrfy') if $m^* \neq m_1, m_2, \ldots, m_q$ and $\mathsf{Vrfy}_k(m^*, t^*) = 1$
- The above is possible under two possible cases:
  - Case I: There exists some $m_i \in \{m_1, \ldots, m_q\}$ such that $h(m_i) = h(m^*)$ --- then $\mathsf{Mac}'_k(m_i) = \mathsf{Mac}'_k(m^*) = t_i$
    - But the probability that $h(m^*) = h(m_i)$ for $m^* \neq m_i$ is negligible ---- as h is a CRHF

Hash-and-MAC Paradigm: Security (Sketch)





- The above construction gives a secure MAC for arbitrary-length messages

PPT Attacker A

I can forge (Mac', Vrfy')

$m_1, m_2, \ldots, m_q$

$t_1, t_2, \ldots, t_q$

$t_i = Mac_k(h(m_i))$

MAC-Oracle

k

Gen(1^n)

Need to formally prove the two cases via suitable reductions

- A successfully forges (Mac', Vr ... $yk(m^*, t^*) = 1$
- The above is possible under two possible ... ses:
  - Case II: There exists no $m_i \in \{m_1, \ldots, m_q\}$ such that $h(m_i) = h(m^*)$
    - Then Vrfyk(m*, t*) = 1 only if A is able to forge Π_{MAC} = (Mac, Vrfy) --- contradiction

Key Management/Agreement

How do Parties Maintain Keys ?

- Several ways depending on the applications
  - Personally meeting and agreeing on several keys
    - Ex: several keys embedded in a secure
    - Common in military application
  - Use some "secure courier" service

> Assumption: Secure channel available at some point

- Depend on a trusted key-distribution center (KDC)
  - Used in large "closed" organizations, ex a University, a company, etc
  - Several practical protocols ba
    - Ex: Needham
    - Forms the b … networked

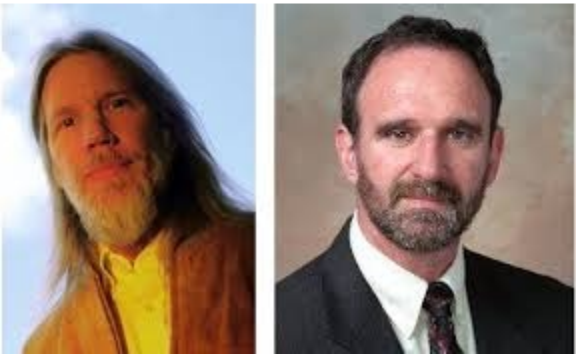
> Assumption: Secure channel available … point + Trust on KDC + … possibility for Single- … point-failure

Diffie-Hellman Key-exchange protocol

- Birth of the public-key revolution

- Can parties establish secure … on a public channel without having any prior shared secret ?
- Seems like an impossible task !!

Diffie-Hellman Key Exchange Protocol



Whitfield Diffie and Martin Hellman. New Directions in Cryptography. 1976

- Showed how two people can publicly establish a secret-key even if an eavesdropper monitors the entire conversation

- Underlying observation: asymmetry is often present in the world !!

No key required

Not possible

Based on some assumptions in (some) cyclic groups of prime order

Extremely difficult

Roadmap

Groups

↓

Finite groups

↓

Finite cyclic groups

Three Assumptions

↓

Finite Cyclic groups of prime order

(special advantages)

modular arithmetic

Modular Arithmetic

- Central to public-key cryptography
- $\mathbb{Z}$ --- set of integers
- Let a, N $\in \mathbb{Z}$, with N > 1. Then

[a mod N] = remainder when a is divided by N

Proposition: Given a and N, there always exist integers q and r such that :

$a = qN + r$, where $0 \leq r < N$

Notation: r is denoted as [a mod N]

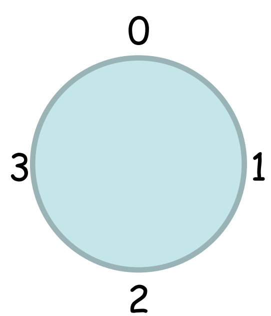
- There exists a unique mapping from a to [a mod N]; f: $\mathbb{Z} \rightarrow \{0,\ldots,N-1\}$

Definition (Reduction modulo N): The process of mapping an integer a to [a mod N] is called reduction modulo N

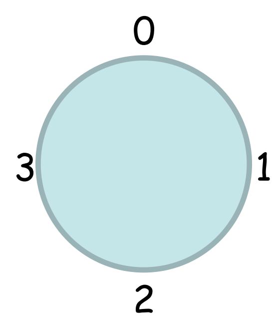
Easy way of Modular Reduction

- To do reduction modulo N, always imagine a clock with marks 0, 1, …, N-1
- Find [a mod N] in the clock notation as follows:
  - If a is positive: start counting from 0 in the clock in a clock-wise direction and stop after counting a times --- the final mark represents [a mod N]
  - If a is negative: start counting from 0 in the clock in an anti clock-wise direction and stop after counting a times --- the final mark represents [a mod N]

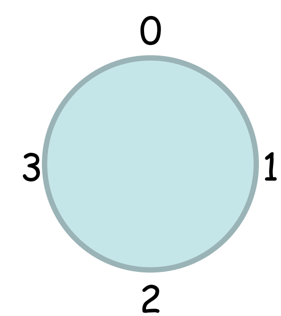
- Ex: N = 4



- [5 mod 4] = 1



- [-7 mod 4] = 1



Congruence Modulo N

Definition (Congruence Modulo N): If [a mod N] = [b mod N], then a is said to be congruent to b modulo N

- a and b are mapped to the same r
- Notation: a = b mod N;
- Note that a = [b mod N] is different; modulo reduction done on b ONLY 36 = 21 mod 15, but 36 =/= 6
- a = b mod N ⇔ N divides (a - b)

Proposition: Congruence modulo N is an equivalence relation: Reflexive, symmetric & transitive

Standard Rules of Arithmetic for Congruence mod N

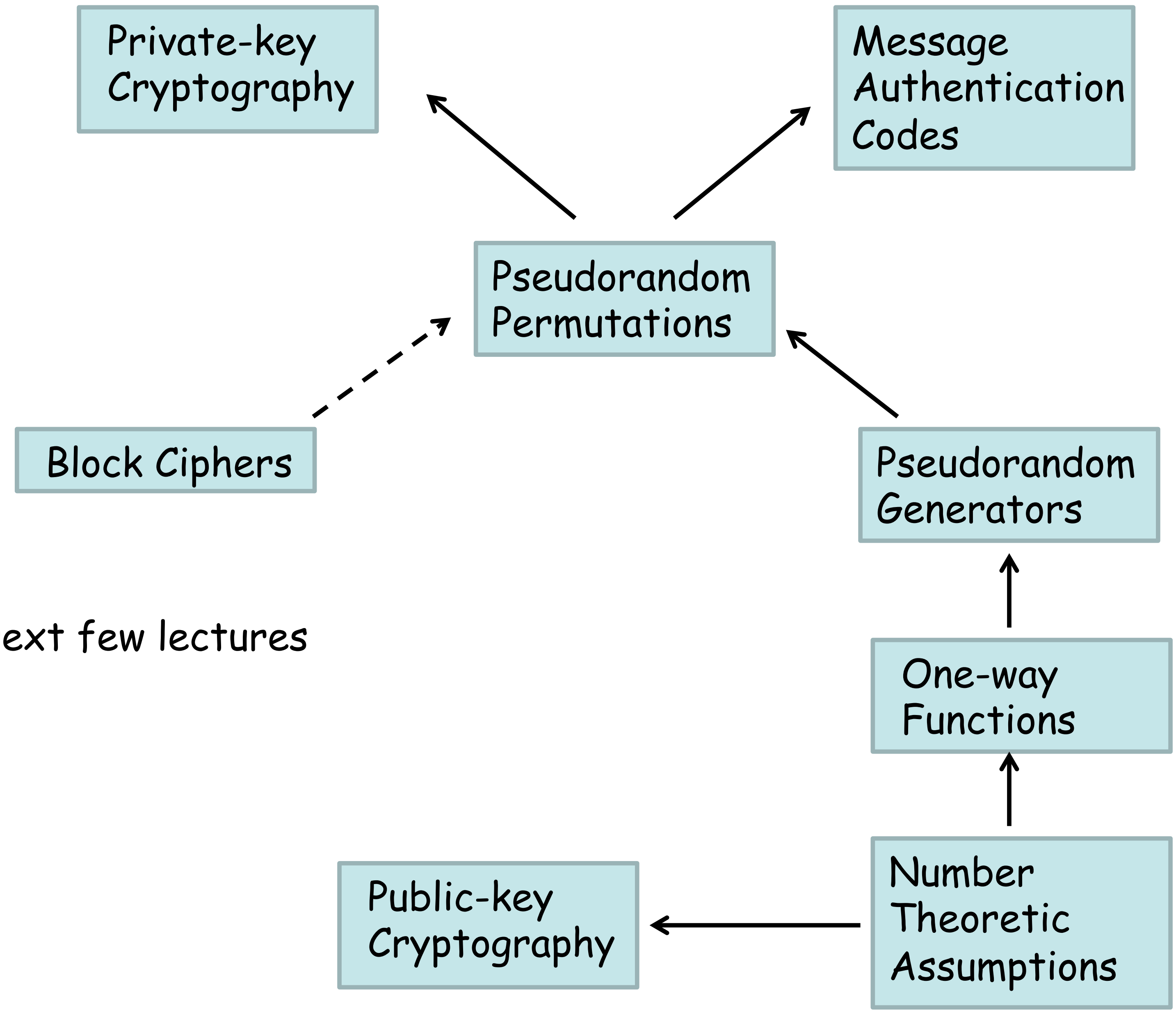
- Yes, trivially for Addition. Subtracti
  - If a = a' mod N and b = b' mod N
  - If a = a' mod N and b = b' mod N then a
  - If a = a' mod N and b = b' mod N then a * b = a' * b' mod N

- Reduce and then add/subtract/multiply
- Instead of add/subtract/multiply and then reduce

- Example: Compute [1093028 * 190301 mod 100]
  - Option I : first compute 1093028 * 190301 and then reduce mod 100
  - Option II : first reduce 1093028 and 190301 mod 100 and get 28 and 1 respectively. Then compute 28* 1 and reduce mod 100
  - Definitely option II is far better than option I

Thank You!

Private-key Cryptography: A Top-down Approach



- Next few lectures