Week 6

Lecture 1

Un Keyed Cryptography: Hash Functions

Lecture 2

Message Authentication Codes or Keyed Hash Function/ Finite Fields

Workshop 3: Workshop based on Lectures in Week5

Quiz 6

Unkeyed Cryptography: Hash Functions

COMP90043

Lecture 1

Udaya Parampalli

Public Key Cryptography: Diffie-Hellman and RSA

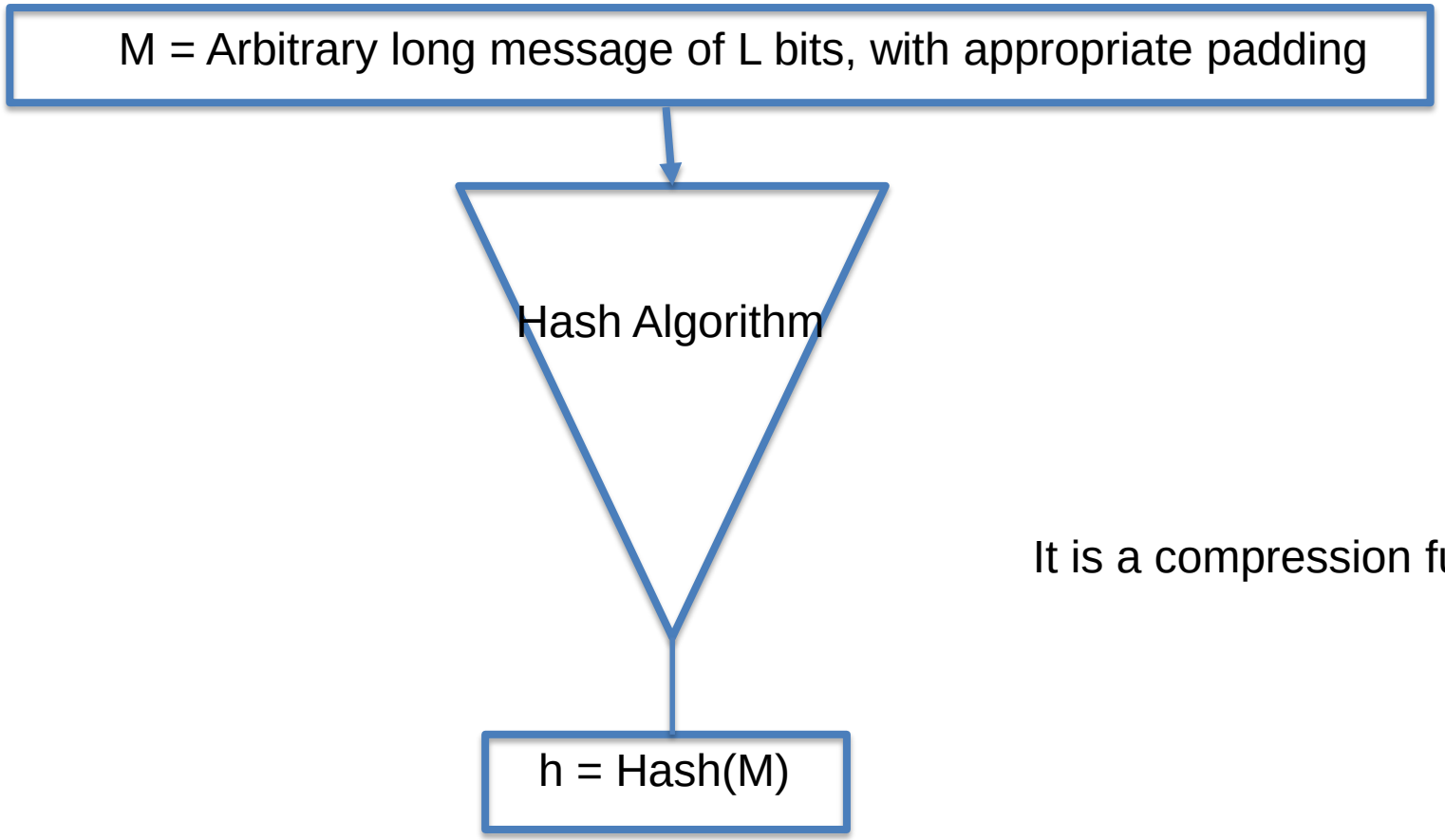
Lecture 1

- 1.1 Preliminary Concepts
  - Hash function and message integrity
  - Uses of Hash functions
- 1.2 Requirements and Clarifications
  - Classification and definitions
  - Requirements
  - Analysis and simple constructions
  - A general construction
- 1.3 Attacks on Hash functions
  - Birthday Attack
  - Block Ciphers as hash functions
  - Practical Hash functions

What is an Hash function

- You would have used hash function while building Hash table data structure, where the purpose is to distribute keys evenly in the Hash table. The function you normally use is ((key) mod a p), a prime number.
- Cryptographic hash function has different requirements.
- In general, the function takes a variable-length data block as input and produces a fixed length tag or digest satisfying certain properties.
- The main objective is to obtain data integrity.
- It is referred as unkeyed primitive as does not require any key.
- As assumed in the other cryptographic functions, the definition of Hash function is also public.
- Hash is also referred to as message digest.

Cryptographic Hash Function



It is a compression function

Integrity

- How does Hash function provide Integrity?
- They are called as Modification Detection Codes (MDC).
- The function Hash has a property that a small change in the message introduces unpredictable changes in the hash value, h = Hash (M).
- If a message is changed while in transit, then running Hash function at the received message tells you how the value is deviated from the hash value computed at the source, thus assuring integrity with high probability.
- You would have studied CRC (Cyclic Redundancy Check) in the data networks. For integrity objective, they provide similar guarantees. The size of the CRC hash is small 32 bits.
- Cryptographic hash is much longer than CRC but also provide except that that it has many more properties, which we will study here.

Integrity in Active Setting.

- Can Hash functions provide Integrity in Active Setting?
- Alice Malice Bob
- Hash(M) = h1
- < M, h1> → Modify M to M’
- Hash(M’) = h2
- < M’, h2> →
- Verify Hash(M’) =? h2
  Accept(M’)

Any intermediate adversary could change the data and compute the hash again. Thus the modification done by Malice cannot be detected.

As usual, the algorithm for the Hash function is public.

Hash function for Authentication?

- A simple answer is “No”. However, understanding of it requires some serious introspections.
- When used without a secret, it can best detect integrity violations due to natural errors that might occur during the message transmission.
- In active setting, an adversary can recompute hash for the things that he/she changed.
- To provide authentication with Hash function, you need to involve other encryption techniques, for example, Alice and Bob need to protect the hash value using encryption techniques.
- We explain issues with the scenario shown in Stallings Fig 11.3 and 11.4.

Hash for Message Authentication

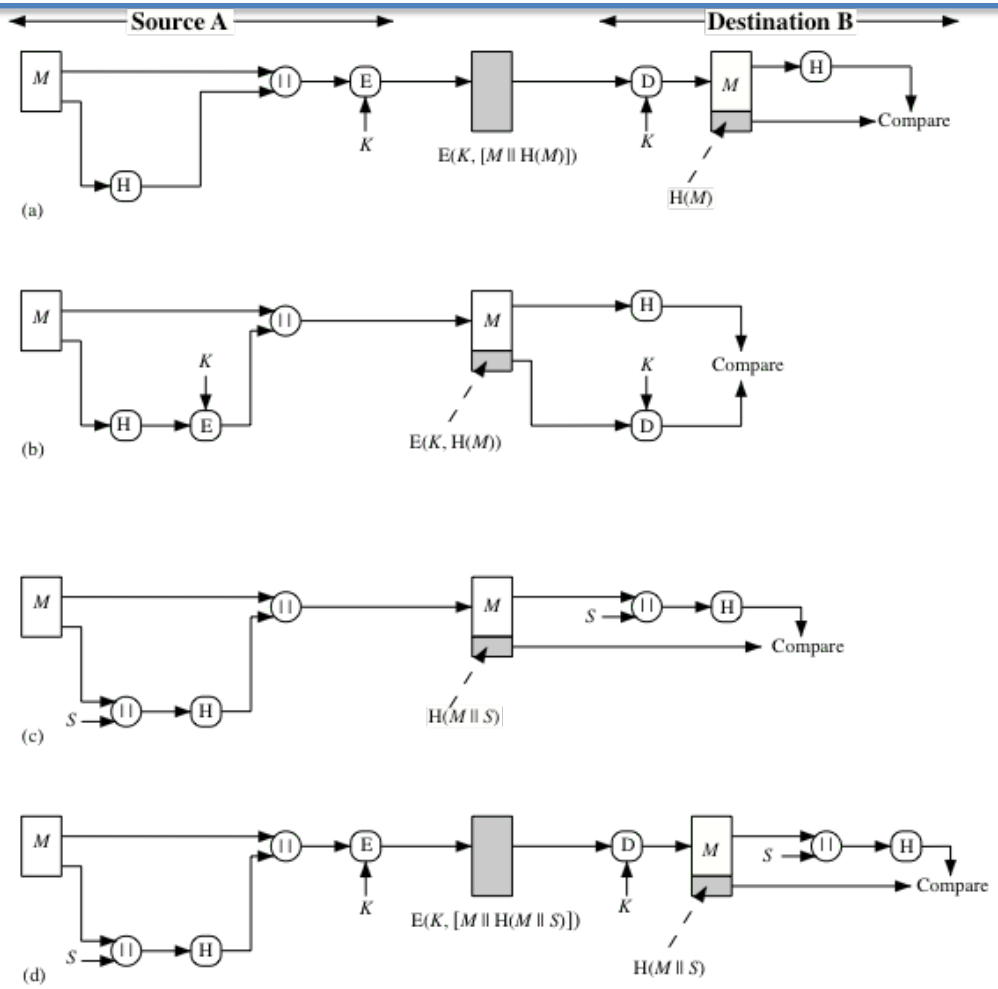
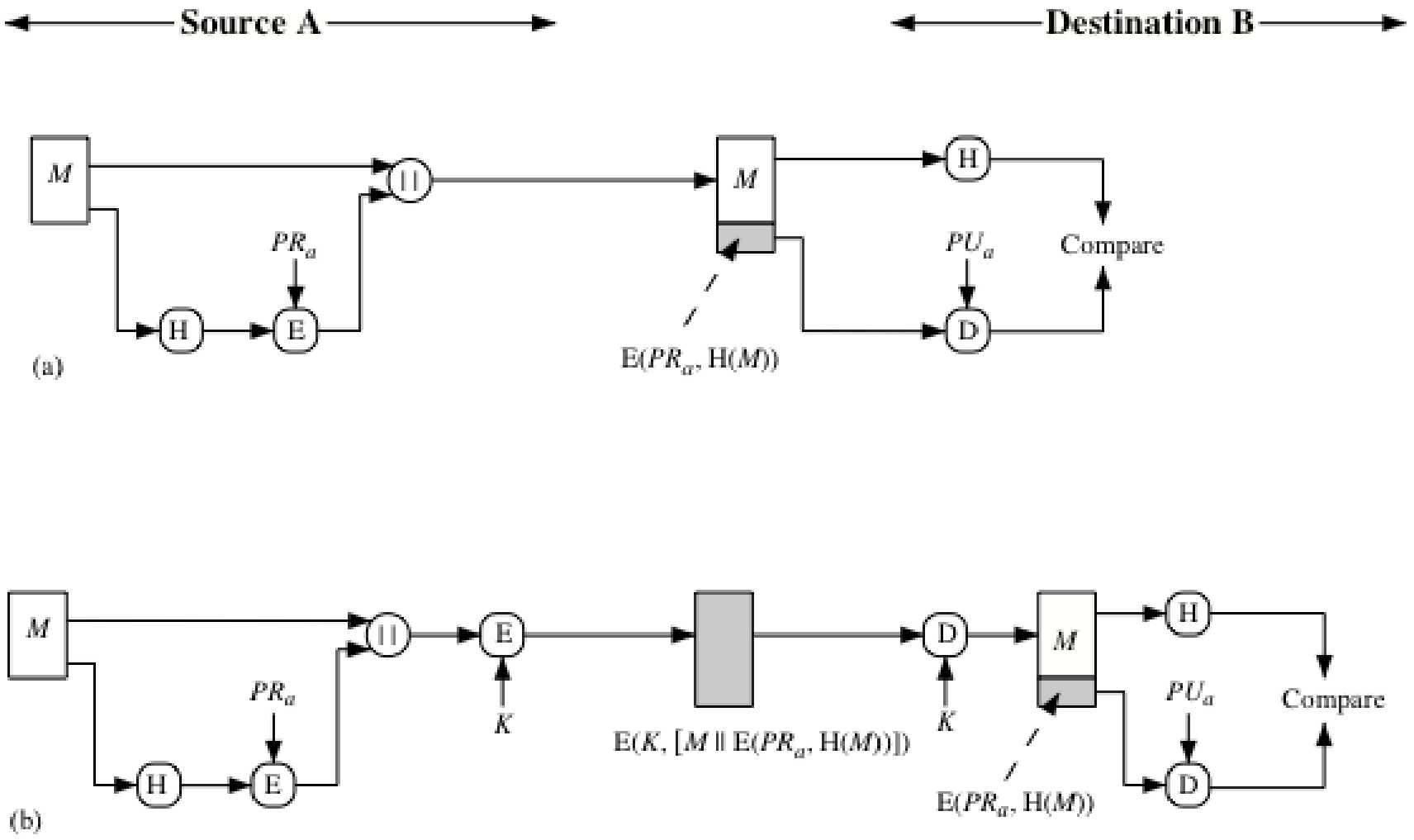


Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication

Use with Digital Signature



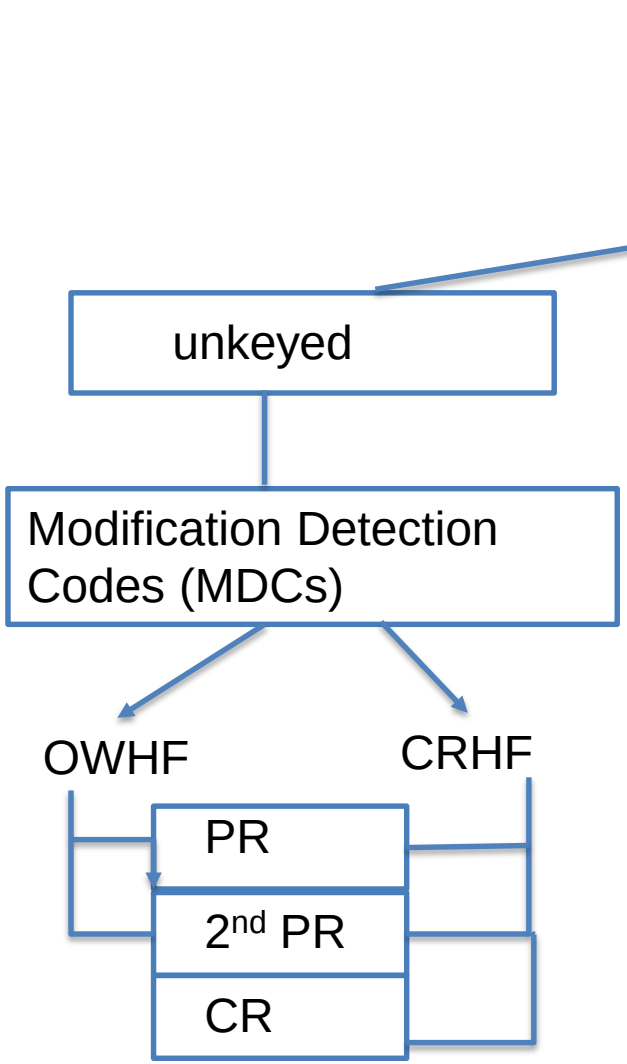
Source: WilliamStallings, Cryptograph and Security

Digital Signatures

- Hash functions are essential to Digital Signatures.
- Here, a hash value is encrypted using public key of the receiver, so the receiver with private key can decrypt and verify the hash value, there by providing message authentication.
- If you need origin authentication, then the hash value need to be signed as well.
- Please read the textbook for studying other uses of hash functions.
- Signcryption is another public key primitive which does these two operations in one go.
- Some of these issues are the topic of discussion in later part of the course.

- Let us consider the classification of Hash functions, next to understand the issues.

Hash Functions Classification



We will study this in the second part of this week

Keyed

Message Authentication Codes (MACs)

PR: Preimage resistant
2nd PR: Second PR
CR: Collision Resitance
OWRF: One Way Hash Function
CRHF: Collision Resistant HF

From Handbook of Applied Cryptography

Definitions

- We will assume the domain as message space (arbitrary long bit strings F_2^*) and range as the hash space, generally a fixed binary space (eg F_2^n).
- Hash function we look for is a mapping from F_2^* to F_2^n , n =128, 256, 512etc.
- Hash function should be “easy” to compute.

- “Easy” generally means that an efficiently computable, for example computing with polynomial time complexity.
- “Hard” generally means that computationally infeasible for example having only exponential time complexity.

Definitions

- PR (preimage resistance): Give an element in the hash space, it is computationally infeasible to workout an input in the message space that results the hash value as output. In other words, given y in F_2^n it is not feasible to workout a preimage x' in F_2^* such that H(x') = y.
- 2nd PR (Second preimage resistance): Given a message and hash pair (x, H(x)), it is it is computationally infeasible to workout an another input in the message space that results the hash value as output. In other words, given x in F_2^* it is not feasible workout another preimage x' in F_2^* such that H(x') = H(x).
- CR (Collision resistance): It is computationally infeasible to find two different input messages in the message space that results in the same hash value as output. In other words, it is not feasible workout two messages x and x' such that H(x') = H(x).

Hash Function Requirements

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness

Table 11.1 from the textbook

Hash Function Relationships

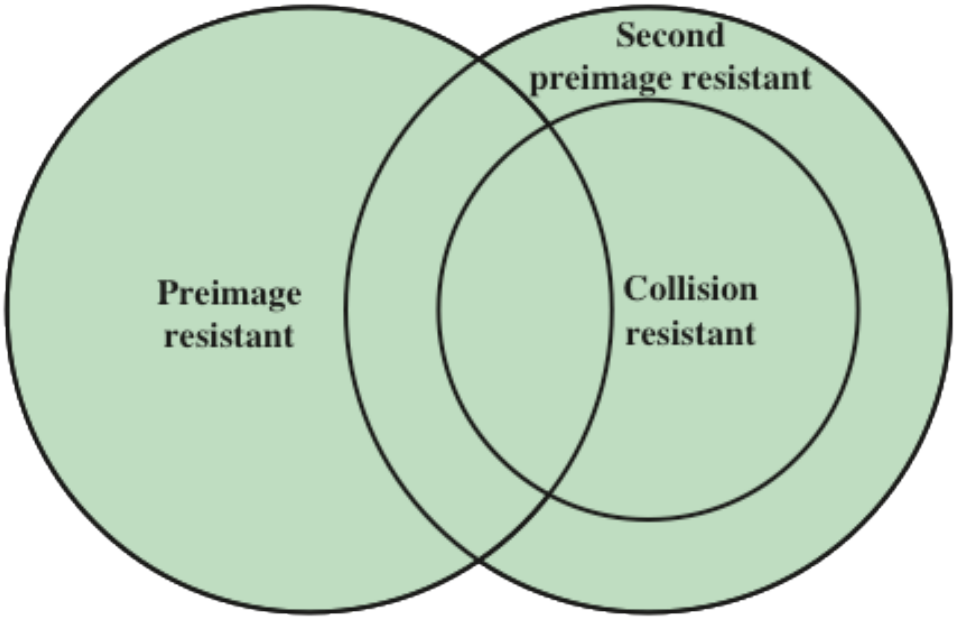


Figure 11.6 Relationship Among Hash Function Properties

Fig 11.6 from the textbook

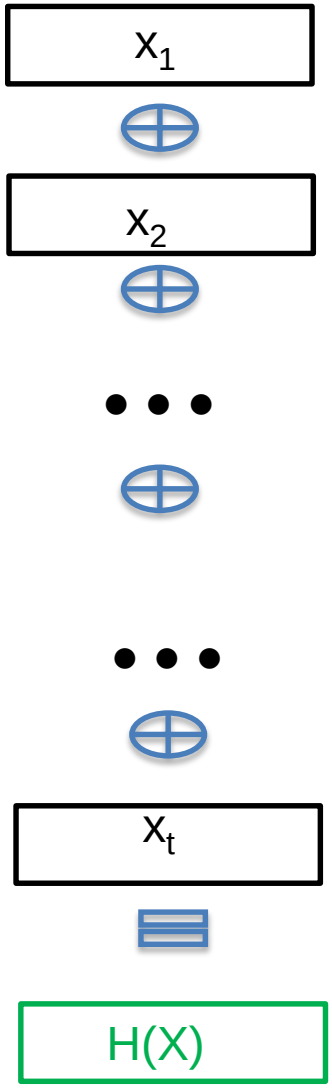
Analysis

- CR implies Second Pre Image Resistance(Second PIR).
  - Let H be CR but not PIR, then
  - Given m, H(m), one can find a second pre image of h(m) as m'. Then you discovered m and m' being the same hash contradicting the fact H is CR. So this situation cannot happen, So CR implies Second PIR
- However, the converse is not necessarily true.

How to Construct Hash functions?

- Main requirements, the function should work for arbitrary input, i.e. F_2^* .
- It should have a fixed length output, of length n, i.e F_2^n.
- One way to arrange input as sequence of n bit blocks, in that case if the input is not multiple of n then add sufficient padding.
- Let [$X = x_1 \| x_2 \| \ldots \| x_t$], and Hash defined as $H(X) = x_1 \oplus x_2 \oplus \ldots. \oplus x_t$
- It can act as MDC, this is a simple parity check code.

- But is this Hash secure?
  - It does not have a preimage resistance, not PR.
  - Consequently it does not have Collision Resistance also, not CR.



An improvement to Parity Check Code

- For the previous construction, make the following changes:
- Add one circular shift after processing each block.
- Let [$X = x_1 \| x_2 \| \ldots \| x_t$], and Hash defined as
- Initialize $h = x_1$
- For each successive block
  - Rotate h to left by 1 bit and xor with the block
- Let T be an left shift operator, then
- $T[x_1(1), x_1(2), \ldots, x_1(n)] = [x_1(2), x_1(3), \ldots, x_1(n), x_1(1)]$

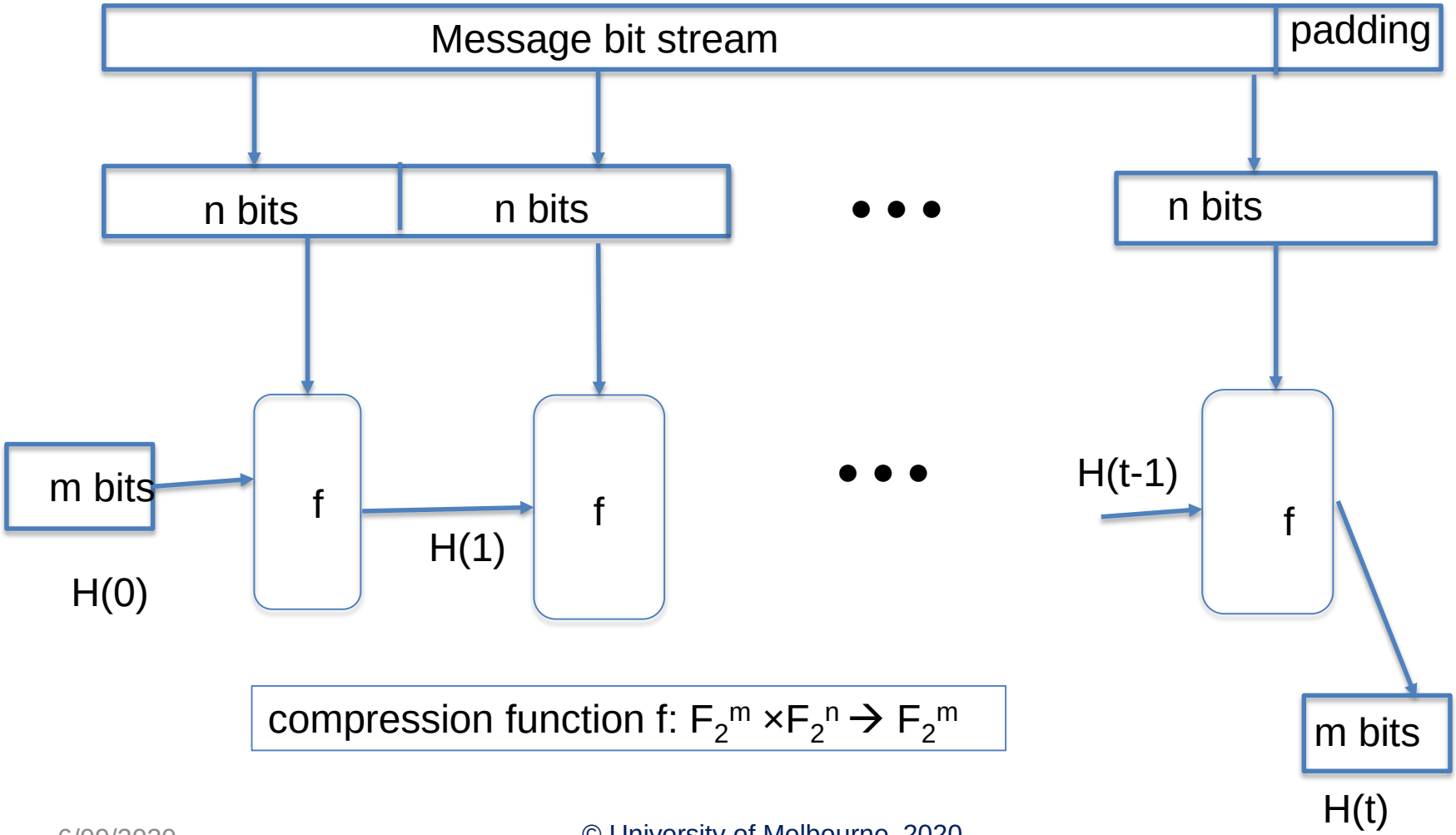
- $H(X) = T(\ldots T(T(T(x_1) \oplus x_2) \oplus x_3) \ldots) \oplus x_t$
- For MDC this is better than the previous PCC,

- But is it good foe security?
- Workout yourself how you can break the PR property.

In general

- Hash Construction methods and Attacks on the constructions goes hand in hand.
- Cryptographers work hard to find good methods; Cryptanalysts will work to break them. It is a competition.
- NIST invited researchers to come up with Hash proposals.
- Having a secure Hash function is still open question for researchers.
- Here I will illustrate a simple method based on the Block cipher constructions mode CBC.
- It is also sometime referred as Merkle–Damgård construction.
- Heart of the construction is a compression function f: $F_2^m \times F_2^n \rightarrow F_2^m$
- Messages are arranged as n bit blocks with possibly the final block padded to make the message length multiple of n.
- The size of the hash is m.

Merkle–Damgård Construction



Attacks on Hash function

- The function should resist brute-force attacks and regular cryptanalysis.
- Attack against PR: Given a random hash value, determine y such that H(y) equals to the hash value.
- Attack against CR: The task is to determine any two messages whose hashes are same, i.e determine x, y such that H(x) = H(y).
- What should be the size of hash value?

- Next we will illustrate the Birthday attack and its implications on the size.

What is the Birthday Attack?

- Consider a set of events of N different outcomes (think of this as numbers from 1 to N). Assume that they are equally likely. P(i) = 1/N for all i;
- Question: What is the probability that after n trials at least two of the outcomes will be the same?

- Example: Consider 128 bit hash values that are used in the digital signature algorithm for messages of size 1000 bits. There are 2^{1000} possibilities for the messages. The hash values are of length 128 bits and it is drawn uniformly from the space of 0; 1; 2^{128}; N here is 2^{128}.
- How many messages you draw from the message space such that any two of them have the same hash?
- Is it 2^{128} ? The birthday attack tells us you do not have to wait that long, you will see collision with much less messages to the order of 2^{64} !

Birthday Attack

- This is not a surprizing result, it follows from elementary probability theory.

- Consider random numbers from 1 to N using uniform distribution.
- Then, the probability that any two numbers are same exceeds 0.5 after roughly about Square root of N trials. [Please read the textbook]

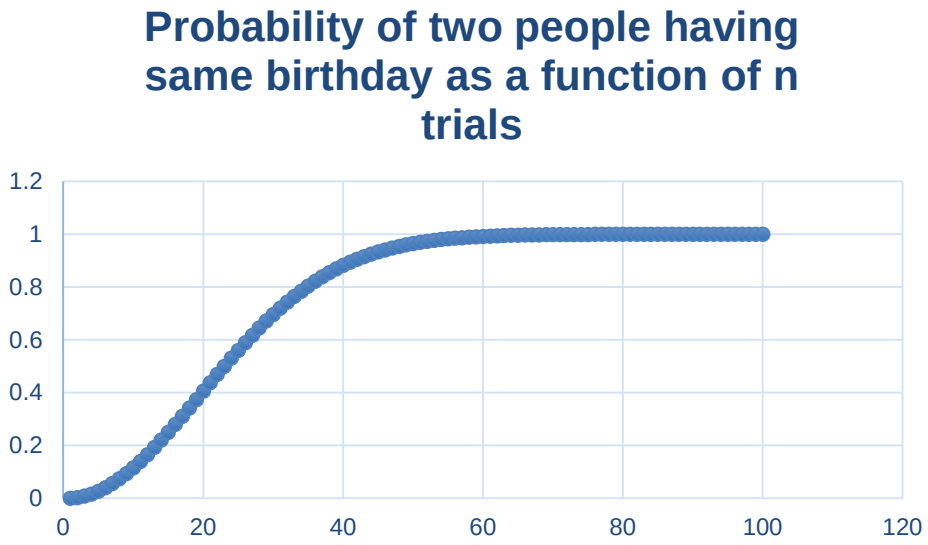
- So for m bit hash function, there are 2^m possible hash values, Squareroot of $2^m = 2^{(m/2)}$.
- Thus, $2^{(m/2)}$determines strength of hash code

Birthday Atatck

From Probability theory, The probability that after n trials at least two outcomes will be the same is at least ε

$$1 - e^{\frac{-1(n-1)n}{2N}}.$$

There are 365 days, N = 365, so plot of ε as a function n (trials) is below;
You only need around 23 trials before you see a collison!



Magma code:

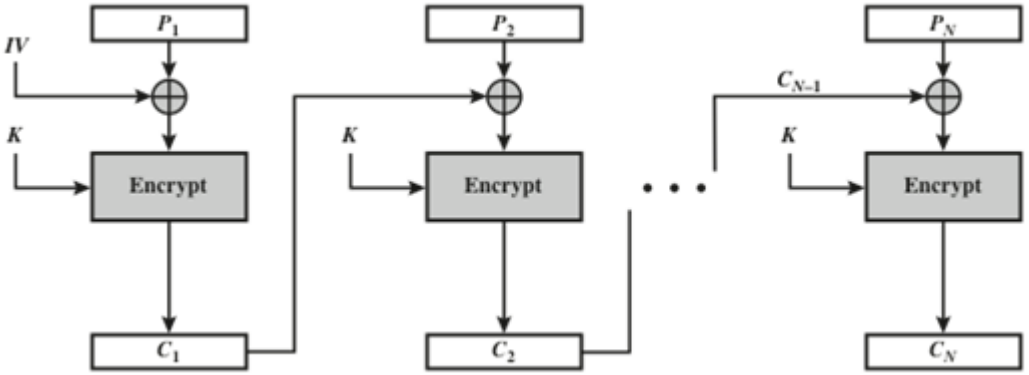
```
for n:= 1 to 100 do
[n, 1- Exp( -(n*(n-1))/(2*N))];
end for;
```

An example from text

- Yuval's strategy of using the above attack for creating collisions.
- Please read 11.3 of the textbook.
- In a scenario, a given user is prepared to sign a valid message x
- An adversary generates $2^{(m/2)}$ variations x' of x, all with essentially the same meaning, and saves them
- The adversary then generates $2^{(m/2)}$ variations y' of a desired fraudulent message y
- Notice that the two sets of messages are compared to find pair with same hash, you will find them with probability > 0.5 by the attack.
- Now adversary can have user sign the valid message, then in the signature replace the corresponding message that resulted in collision, you have a forgery!
- What do we learn from this? The size of the hash should be large enough that finding collisions are impracticable. You need 128 to 512 bits for hash in practice.

Block Ciphers as Hash functions

- You can use Block ciphers in CBC mode to create hash functions.
  - Initialize H_0=0 and zero-pad of final block
  - compute: $H_i = E_{M_i} [H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key
- If DES function is used the resulting hash is too small (64-bit) and hence vulnerable to birthday attack.
- Also, there is a variant of birthday attack called “meet in the middle”, please go through the textbook.



CBC mode encryption
Fig from the textbook

More practical Hash functions

- Secure Hash Algorithm is a proposal designed by NIST in 1993 and was published as a standard (FIPS180).
- Thee are several revisions to this and a lot of activities happened in breaking these schemes. SHA-1's CR property was broken!
- The latest standard is SHA-3 which is considered to be safe.

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Table 11.3 from the textbook

Week 6

Lecture 1

Un Keyed Cryptography: Hash Functions

Lecture 2

Message Authentication Codes or Keyed Hash Function/ Finite Fields

Workshop 3: Workshop based on Lectures in Week5

Quiz 6