CRYPTOGRAPHIC HASH FUNCTIONS

LECTURE 11

Cryptography

Hash Functions

- Condenses arbitrary message to fixed size

$$h = \mathrm{H}(M)$$

- Hash used to detect changes to message
- Want a cryptographic hash function
  - computationally infeasible to find data mapping to specific hash (one-way property)
  - computationally infeasible to find two data to same hash (collision-free property)

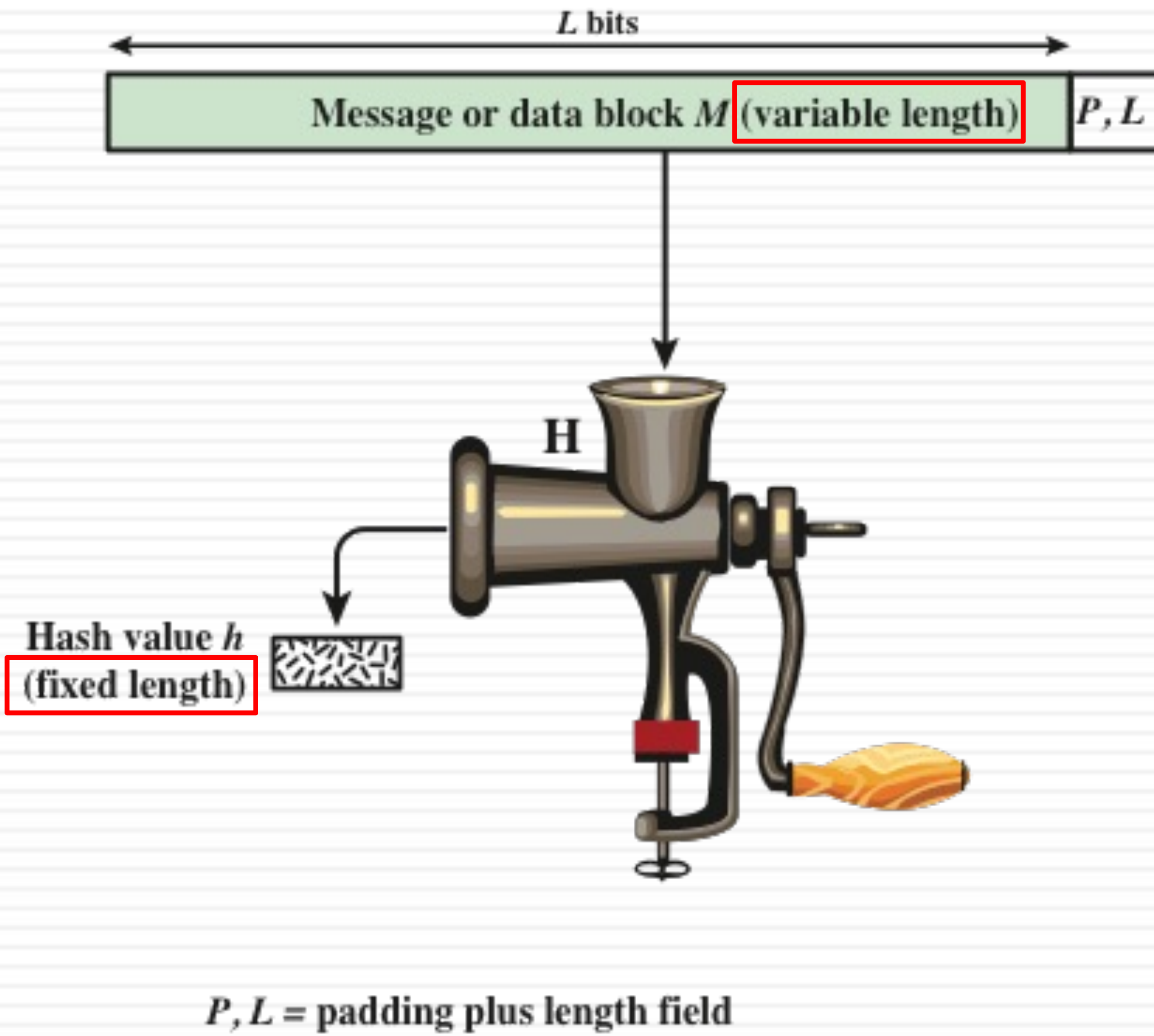


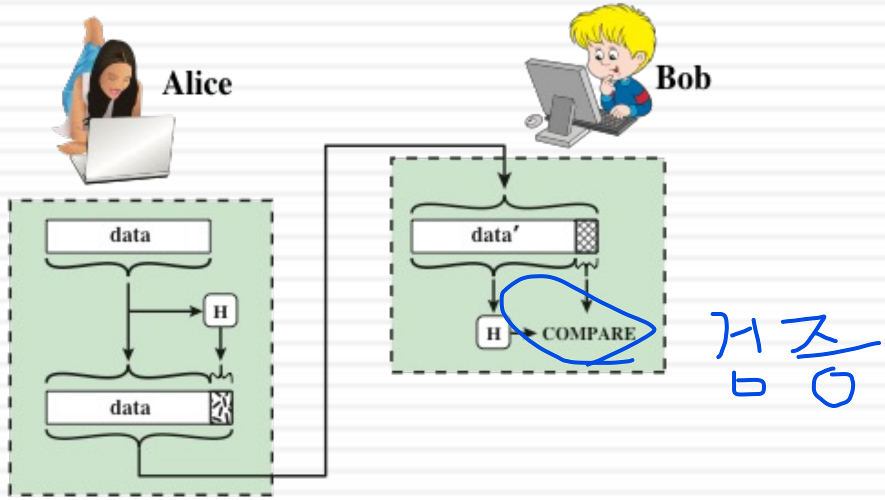
Figure 11.1 Cryptographic Hash Function; $h = H(M)$

Examples 1

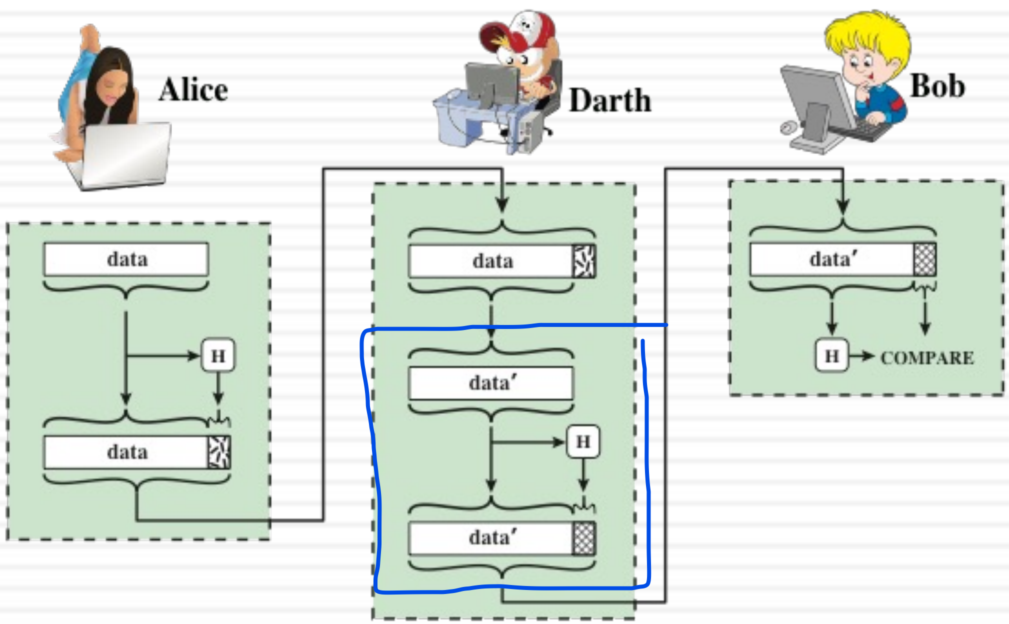
- MD5(“산에는 꽃이 피네 꽃이 피네”) = cecedb0184e9e0ef5b054a008330bad9
- SHA1(“갈봄 여름 없이 꽃이 피네”) = d615020eb9bf5ae7ff5616f8c879cf3f18a79b0b
- SHA256(“산에 산에 피는 꽃은”) = 1a44eed17ff31cf8c75edcfb96f4925c0b79286b950889feab911b9c250d7b48
- SHA384(“저만치 혼자서 피어 있네”) = bf1090a8f3241f160184f91a0134108c5d285f46ba6e830536744dd85a07f26b03e986e60ff921384d2133572f178948

Examples 2

- SHA1("산에서 누는 작은 새여") = 3fb623d4b169983a6a4b9714d9881a52e6a13848
- SHA1("산에서 누는 작은 새여,") = 3d0f0499aeb9520ff87096b8413af492a3c0e50b
- SHA1("?") = 216f3e1d80110cb490a93a83fcaaa25bcdf5248e
- SHA1(비트코인거래내역:?) = ~~1d6720c6b72dde3666c4ce7def628fda46547~~7b32



(a) Use of hash function to check data integrity

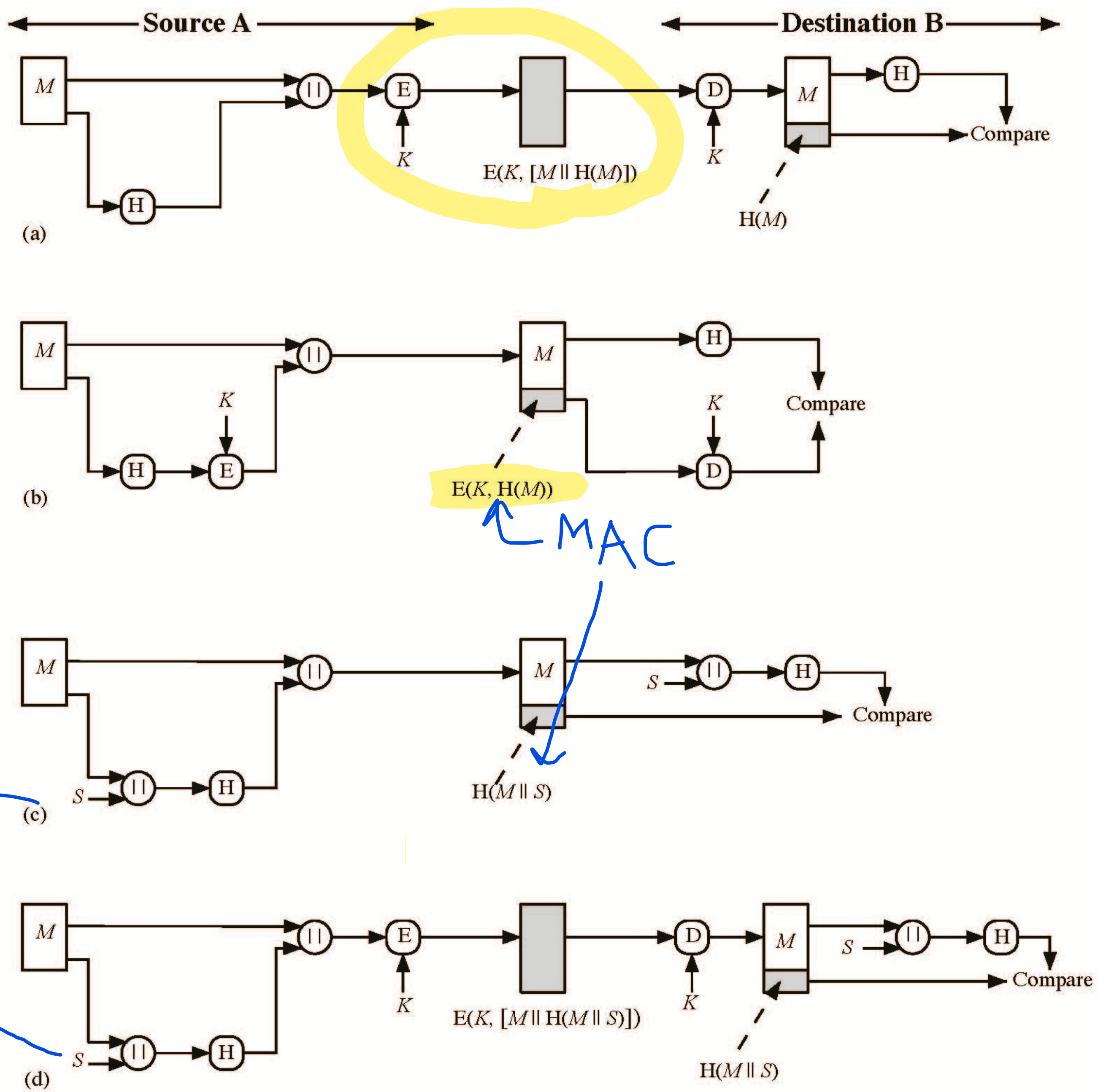


(b) Man-in-the-middle attack

Figure 11.2 Attack Against Hash Function

Hash Functions & Message Authentication

공유 비밀값



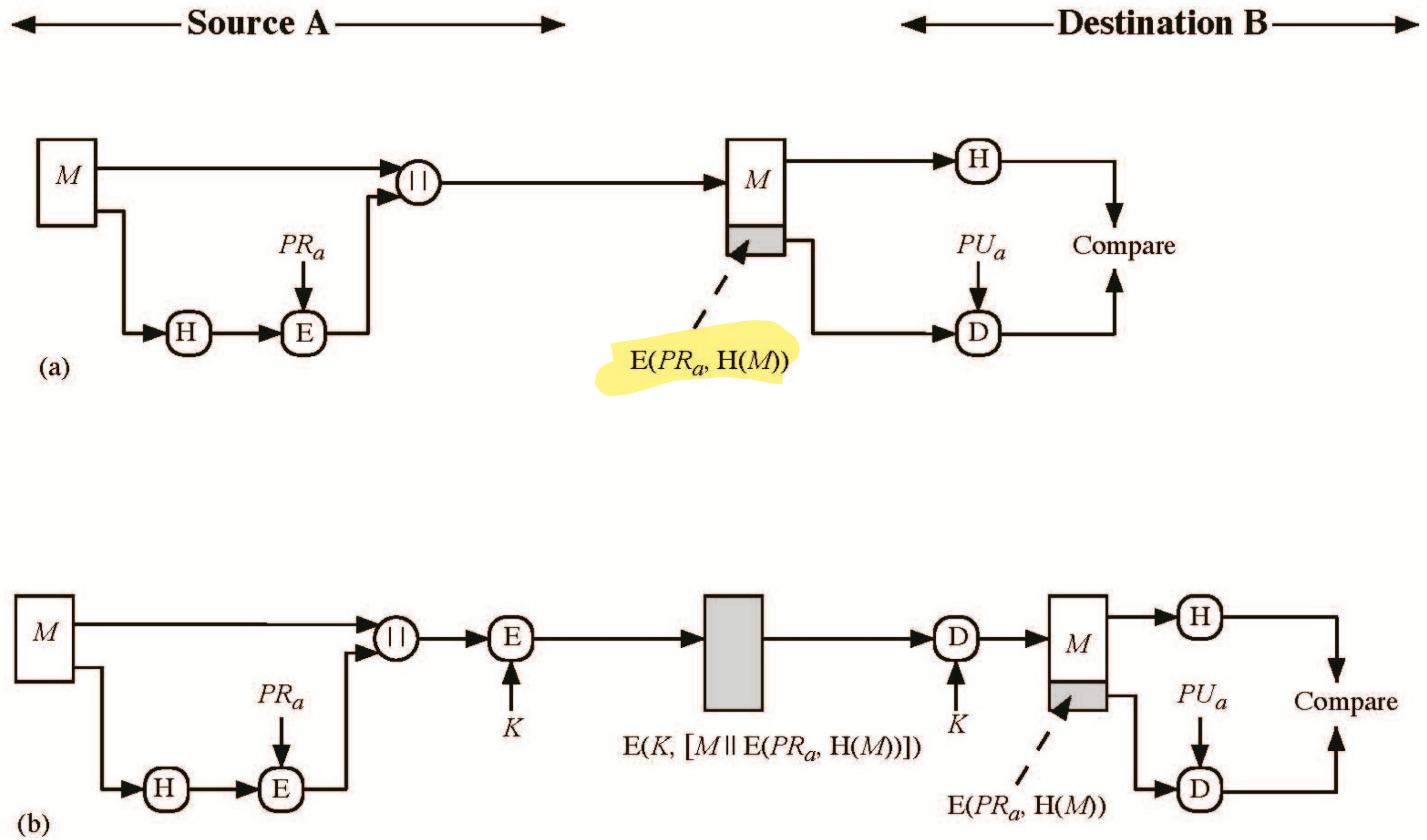
Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties

Digital Signature

- Operation is similar to that of the MAC
- The hash value of a message is encrypted with a user's private key
- Anyone who knows the user's public key can verify the integrity of the message
- An attacker who wishes to alter the message would need to know the user's private key
- Implications of digital signatures go beyond just message authentication

Digital Signatures



Other Hash Function Uses

- To create a one-way password file
  - store hash of password not actual password
- For intrusion detection and virus detection
  - keep & check hash of files on system
- Pseudorandom function (PRF) or pseudorandom number generator (PRNG)

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 ! 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

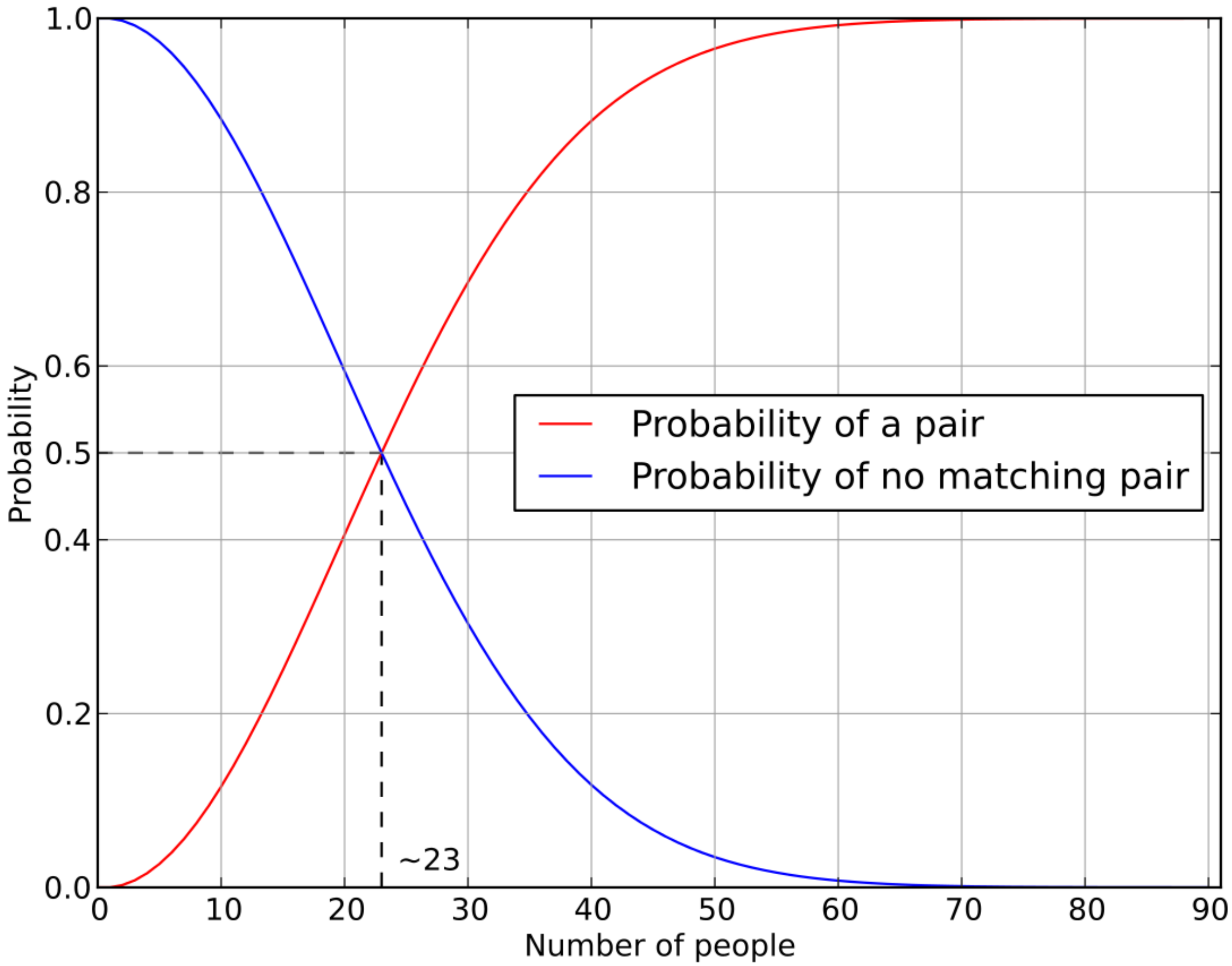
Attacks on Hash Functions

- A preimage or second preimage attack
  - find y s.t. $H(y)$ equals a given hash value
- Collision resistance
  - find two messages x & y with same hash so $H(x) = H(y)$
- The value $2^{m/2}$ determines *strength* of m-bit hash code against brute-force attacks
  - 128-bits inadequate, 160-bits suspect

Birthday Attacks

- Given user prepared to sign a valid message x
- Opponent generates $2^{m/2}$ variations x' of x, all with essentially the same meaning, and saves them
- Opponent generates $2^{m/2}$ variations y' of a desired fraudulent message y
- Two sets of messages are compared to find pair with same hash (probability > 0.5 by *birthday paradox*)

Birthday Paradox



A Letter in 2^{37} Variation

Dear Anthony,

$\begin{Bmatrix}\text{This letter is}\\ \text{I am writing}\end{Bmatrix}$ to introduce $\begin{Bmatrix}\text{you to}\\ \text{to you}\end{Bmatrix}$ $\begin{Bmatrix}\text{Mr.}\\ \text{--}\end{Bmatrix}$ Alfred $\begin{Bmatrix}\text{P.}\\ \text{--}\end{Bmatrix}$ Barton, the $\begin{Bmatrix}\text{new}\\ \text{newly appointed}\end{Bmatrix}$ $\begin{Bmatrix}\text{chief}\\ \text{senior}\end{Bmatrix}$ jewellery buyer for $\begin{Bmatrix}\text{our}\\ \text{the}\end{Bmatrix}$ Northern $\begin{Bmatrix}\text{European}\\ \text{Europe}\end{Bmatrix}$ $\begin{Bmatrix}\text{area}\\ \text{division}\end{Bmatrix}$. He $\begin{Bmatrix}\text{will take}\\ \text{has taken}\end{Bmatrix}$ over $\begin{Bmatrix}\text{the}\\ \text{--}\end{Bmatrix}$ responsibility for $\begin{Bmatrix}\text{all}\\ \text{the whole of}\end{Bmatrix}$ our interests in $\begin{Bmatrix}\text{watches and jewellery}\\ \text{jewellery and watches}\end{Bmatrix}$ in the $\begin{Bmatrix}\text{area}\\ \text{region}\end{Bmatrix}$. Please $\begin{Bmatrix}\text{afford}\\ \text{give}\end{Bmatrix}$ him $\begin{Bmatrix}\text{every}\\ \text{all the}\end{Bmatrix}$ help he $\begin{Bmatrix}\text{may need}\\ \text{needs}\end{Bmatrix}$ to $\begin{Bmatrix}\text{seek out}\\ \text{find}\end{Bmatrix}$ the most $\begin{Bmatrix}\text{modern}\\ \text{up to date}\end{Bmatrix}$ lines for the $\begin{Bmatrix}\text{top}\\ \text{high}\end{Bmatrix}$ end of the market. He is $\begin{Bmatrix}\text{empowered}\\ \text{authorized}\end{Bmatrix}$ to receive on our behalf $\begin{Bmatrix}\text{samples}\\ \text{specimens}\end{Bmatrix}$ of the $\begin{Bmatrix}\text{latest}\\ \text{newest}\end{Bmatrix}$ $\begin{Bmatrix}\text{watch and jewellery}\\ \text{jewellery and watch}\end{Bmatrix}$ products, $\begin{Bmatrix}\text{up}\\ \text{subject}\end{Bmatrix}$ to a $\begin{Bmatrix}\text{limit}\\ \text{maximum}\end{Bmatrix}$ of ten thousand dollars. He will $\begin{Bmatrix}\text{carry}\\ \text{hold}\end{Bmatrix}$ a signed copy of this $\begin{Bmatrix}\text{letter}\\ \text{document}\end{Bmatrix}$ as proof of identity. An order with his signature, which is $\begin{Bmatrix}\text{appended}\\ \text{attached}\end{Bmatrix}$ $\begin{Bmatrix}\text{authorizes}\\ \text{allows}\end{Bmatrix}$ you to charge the cost to this company at the $\begin{Bmatrix}\text{above}\\ \text{head office}\end{Bmatrix}$ address. We $\begin{Bmatrix}\text{fully}\\ \text{--}\end{Bmatrix}$ expect that our $\begin{Bmatrix}\text{level}\\ \text{volume}\end{Bmatrix}$ of orders will increase in the $\begin{Bmatrix}\text{following}\\ \text{next}\end{Bmatrix}$ year and $\begin{Bmatrix}\text{trust}\\ \text{hope}\end{Bmatrix}$ that the new appointment will $\begin{Bmatrix}\text{be}\\ \text{prove}\end{Bmatrix}$ $\begin{Bmatrix}\text{advantageous}\\ \text{an advantage}\end{Bmatrix}$ to both our companies.

Secure Hash Algorithm

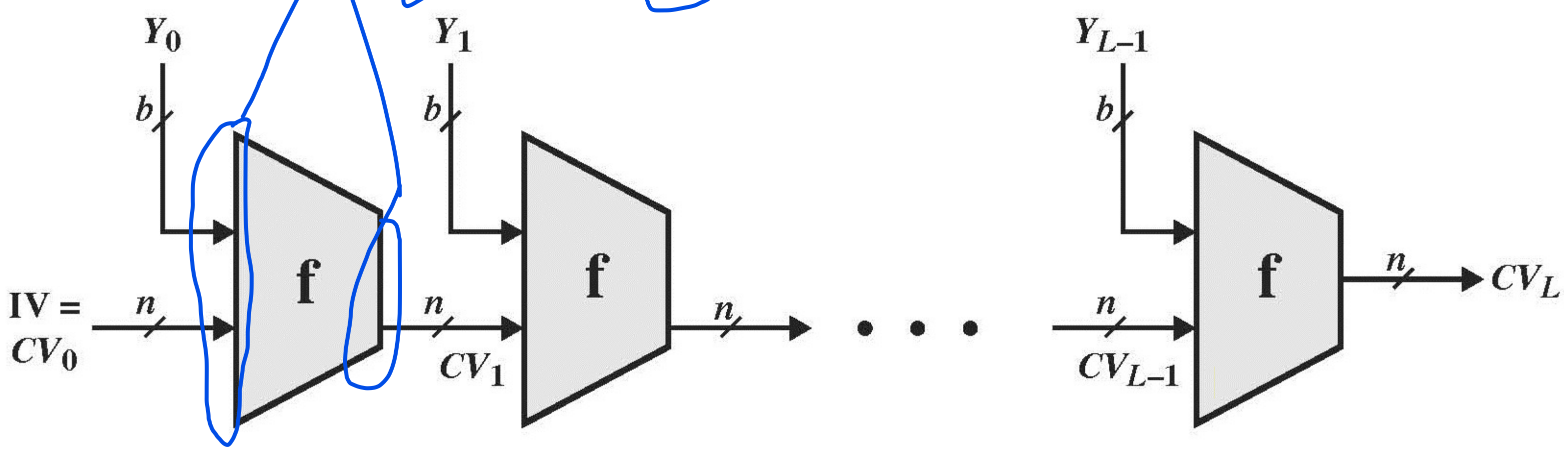
- SHA originally designed by NIST & NSA in 1993
- Was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
- Based on design of MD4 with key differences
- Produces 160-bit hash values
- Recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

SHA-2

- NIST issued revision FIPS 180-2 in 2002
- Adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- Designed for compatibility with increased security provided by the AES cipher
- Structure & detail is similar to SHA-1
- Hence analysis should be similar
- But security levels are rather higher

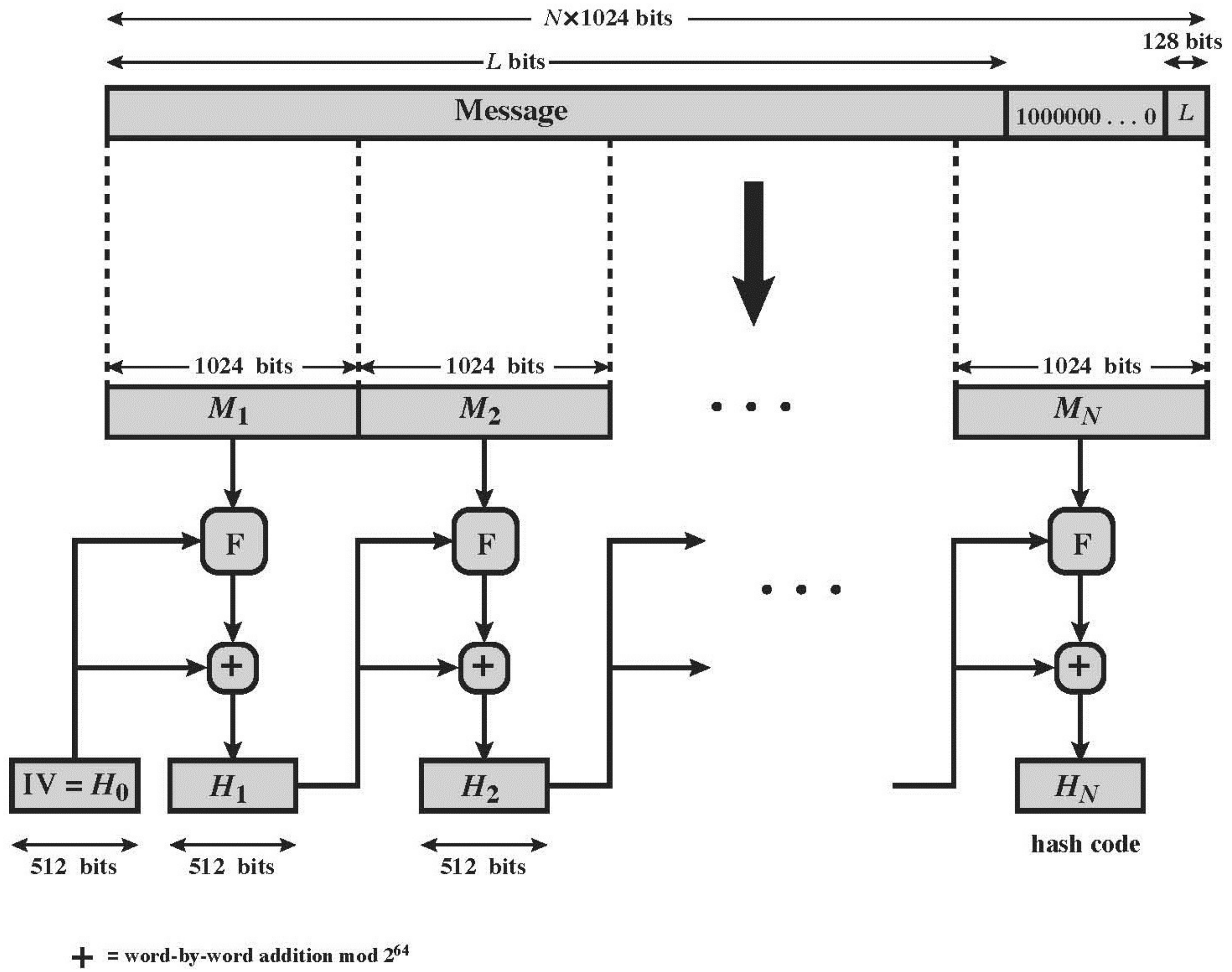
General Structure of Secure Hash

길이다름



IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

SHA-512 Overview



SHA-3

- NIST announced in 2007 a competition for the SHA-3 next generation NIST hash function
- Winning design was announced by NIST in October 2012
- SHA-3 is a cryptographic hash function that is intended to complement SHA-2 as the approved standard for a wide range of applications
- SHA-3 is released by NIST on August 5, 2015

The Sponge Construction

- Underlying structure of SHA-3 is a scheme referred to by its designers as a *sponge construction*
- Takes an input message and partitions it into fixed-size blocks
- Each block is processed in turn with the output of each iteration fed into the next iteration
- The sponge function is defined by three parameters:
  - f = the internal function used to process each input block
  - r = the size in bits of the input blocks, called the *bitrate*
  - pad = the padding algorithm

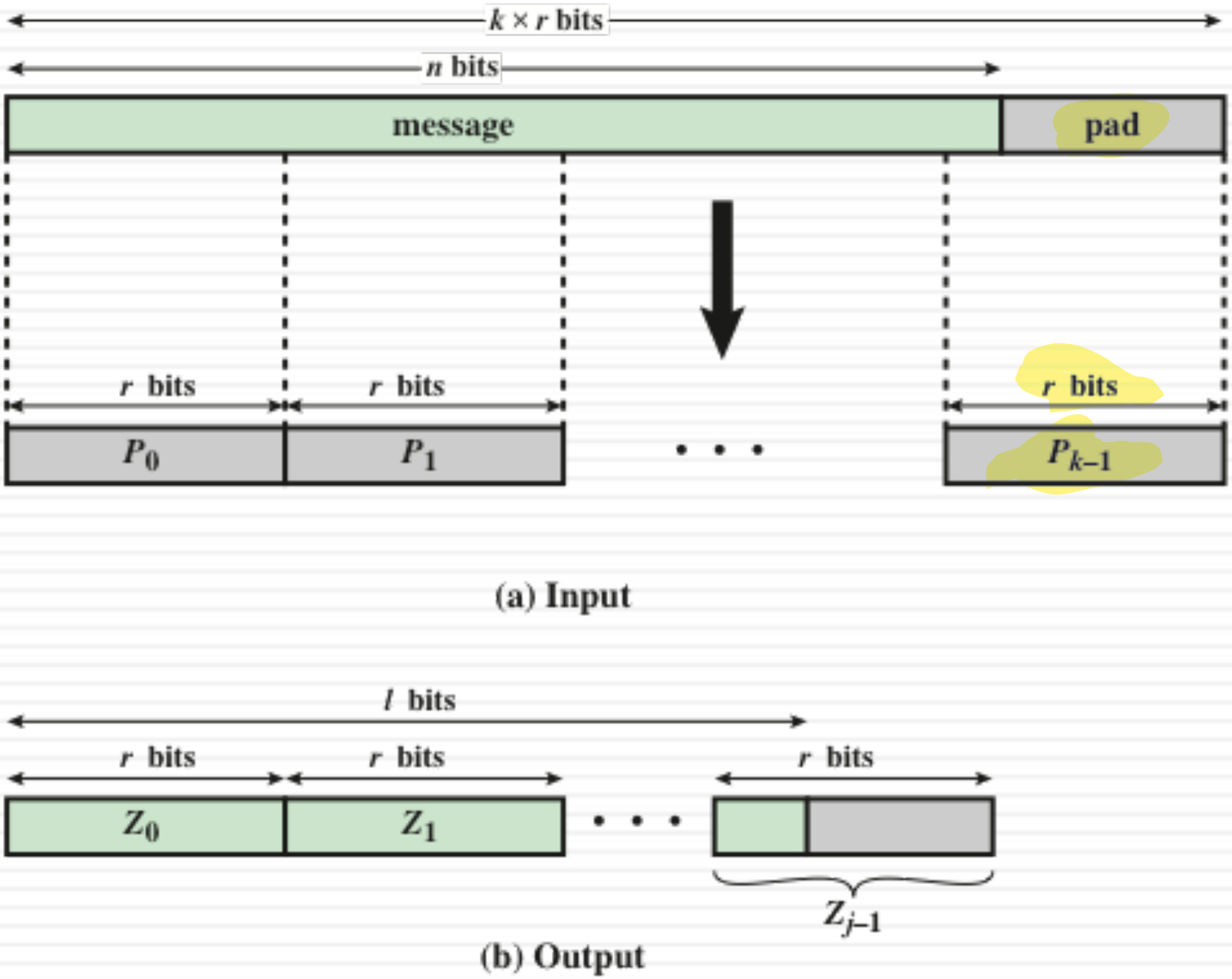


Figure 11.14 Sponge Function Input and Output

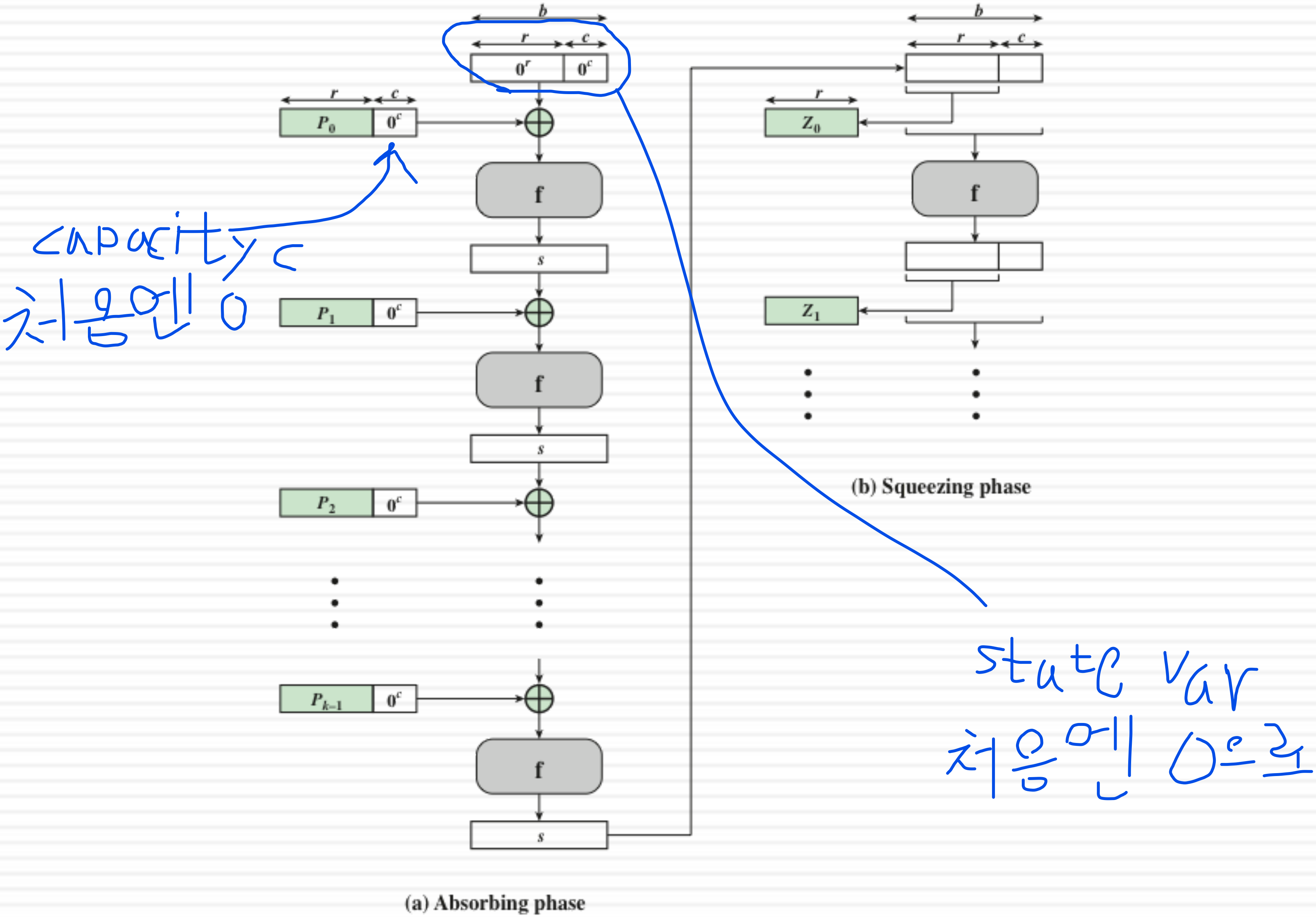


Figure 11.15 Sponge Construction