Cryptographic Tools and Algorithms

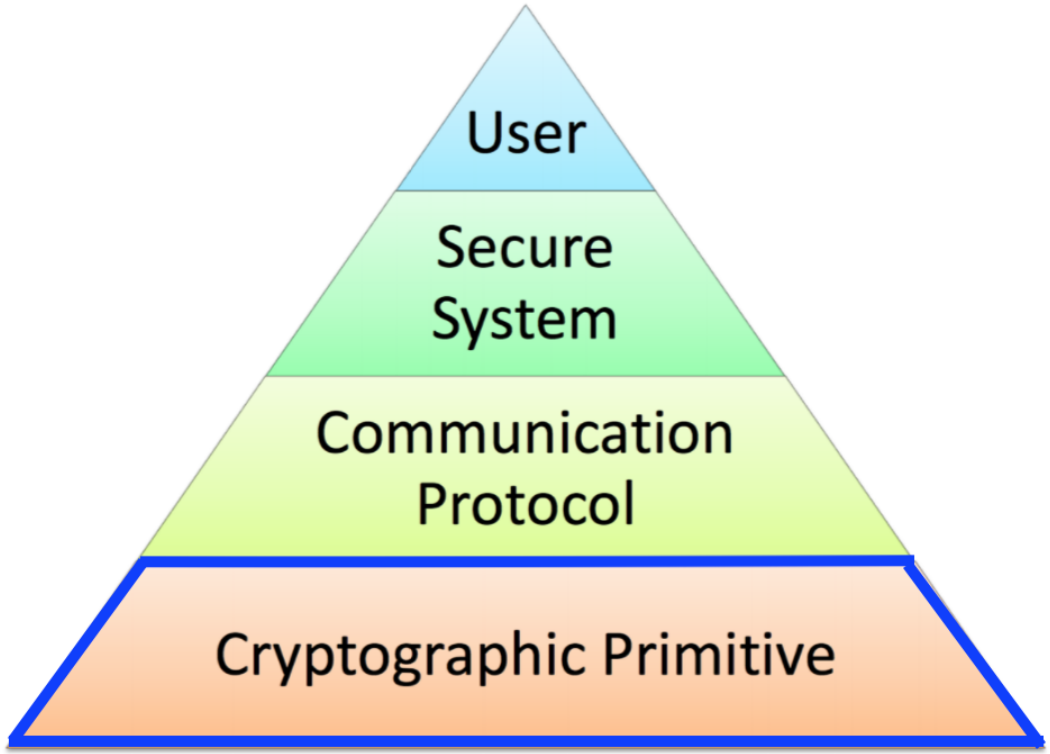
INF140 - Introduction to Cybersecurity

Slides by Chunlei Li[1]

[1] *The slides are adjusted from the ones by Steve Gordon*

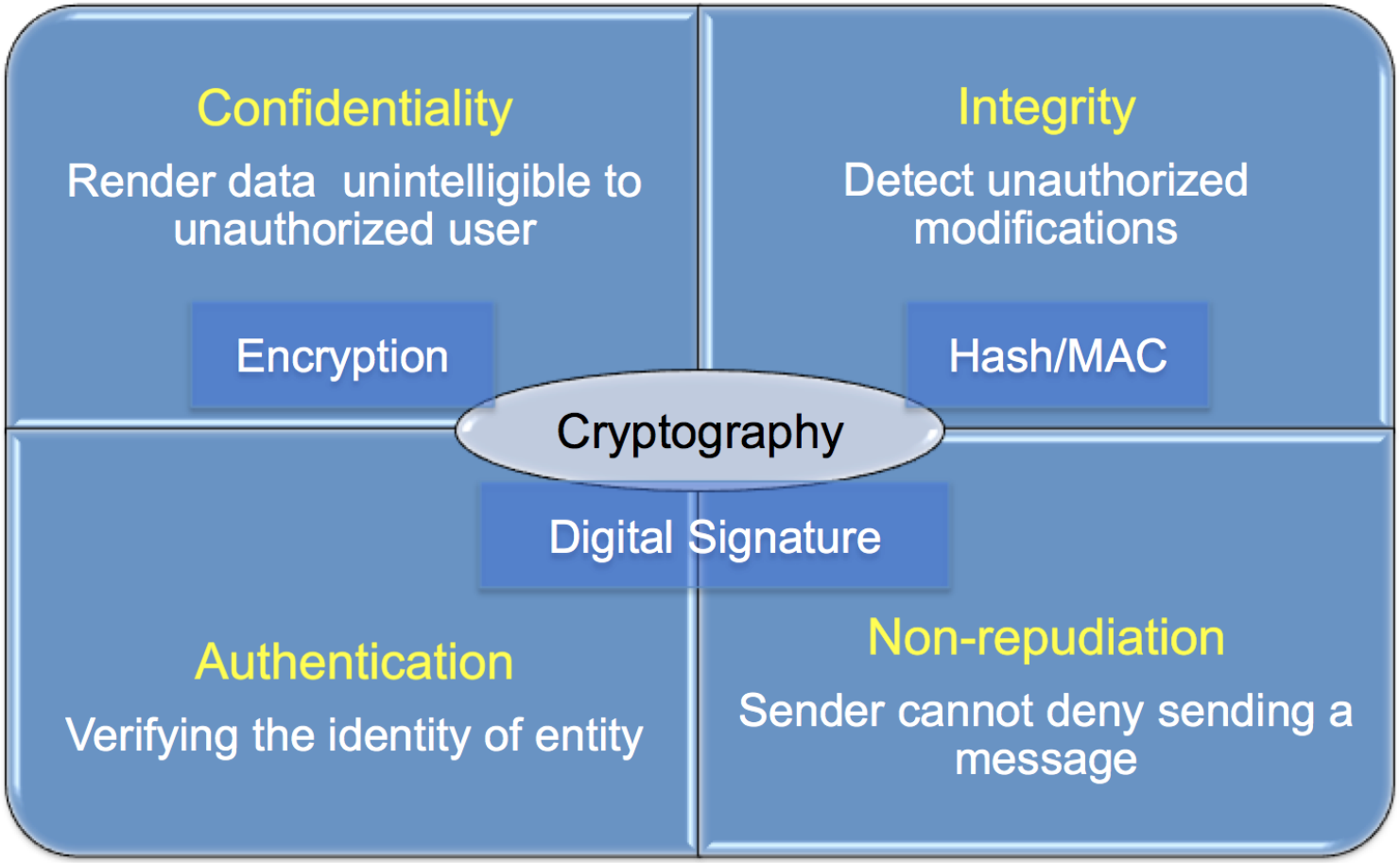
Contents

No Cryptography, No Security!



Cryptography is *necessary but not sufficient* for security

Security Attributes by Cryptography



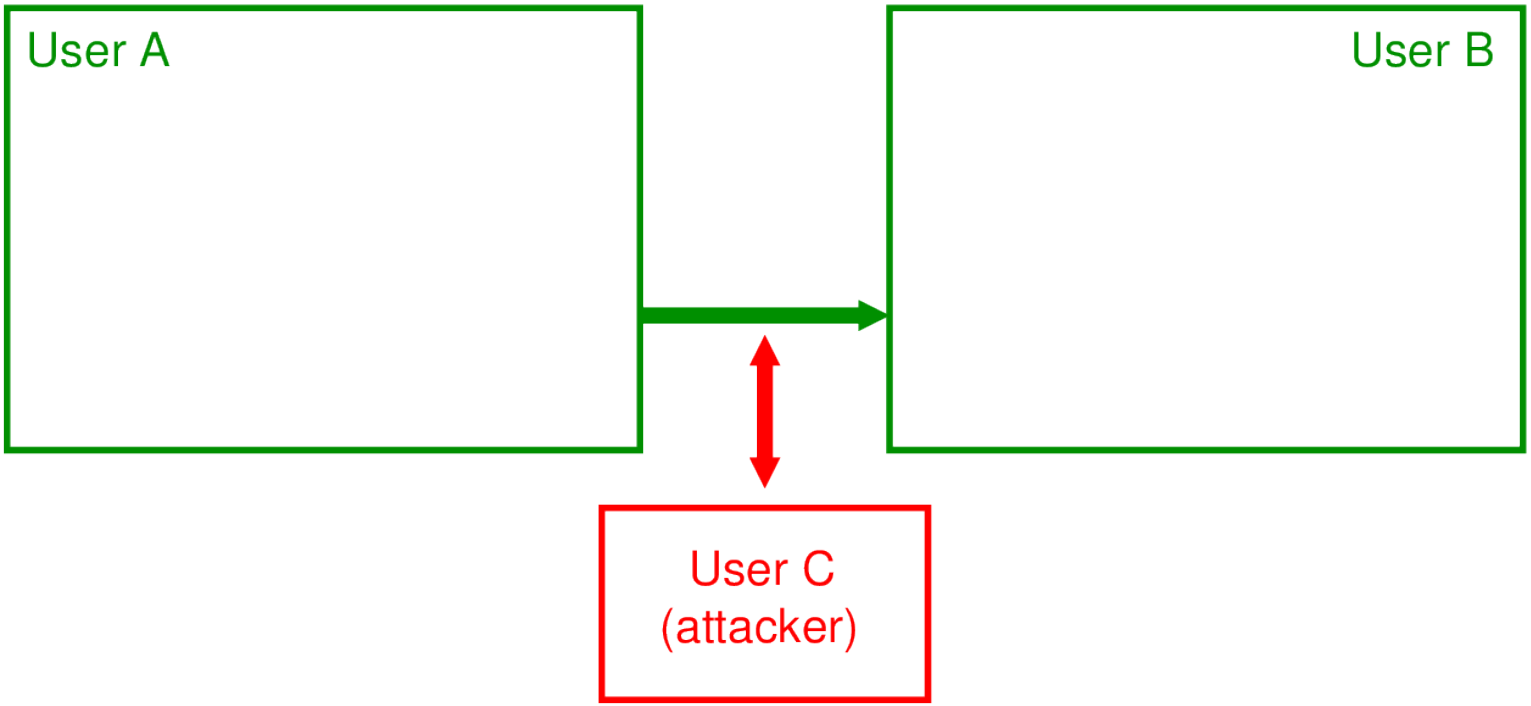
Non-repudiation: one aspect for accountability

Contents

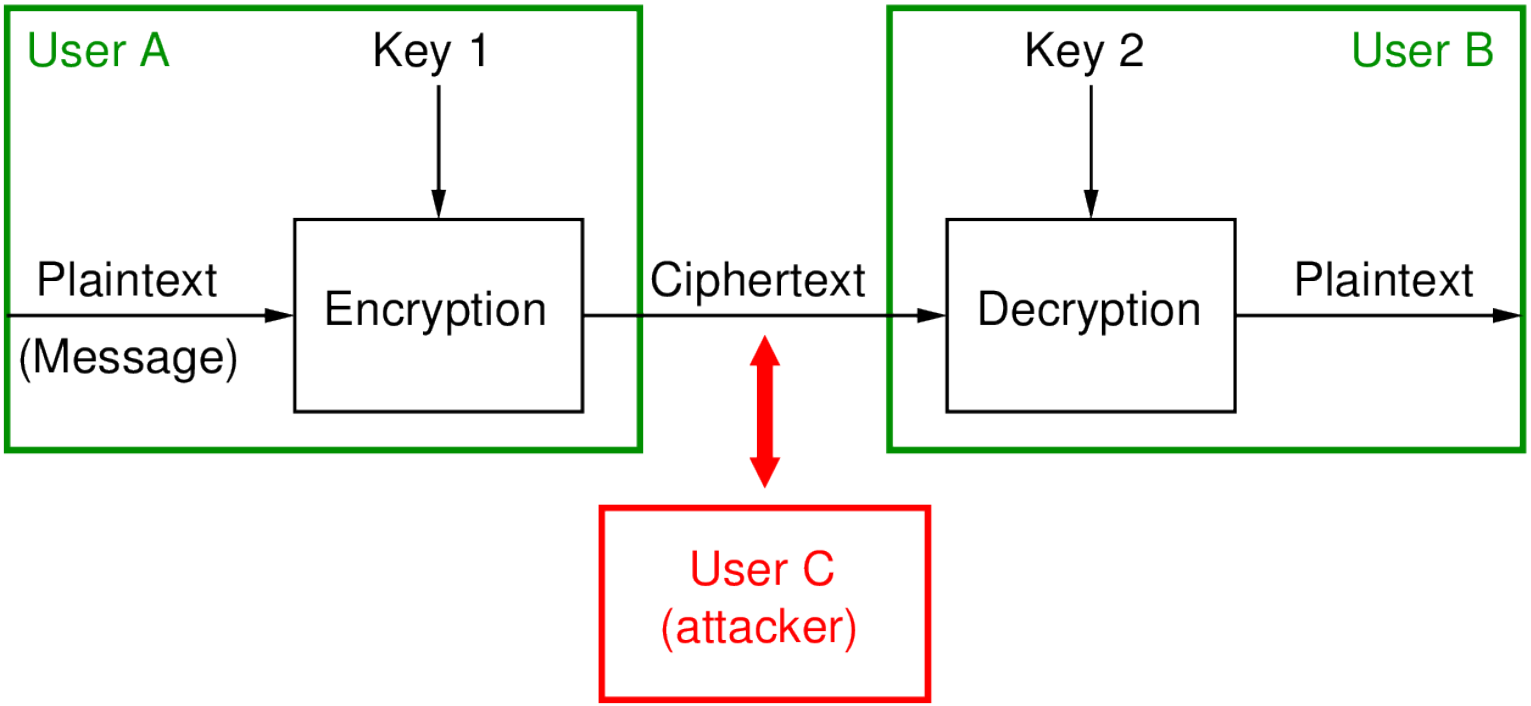
Encryption for Confidentiality

- Aim: assure confidential information not made available to unauthorised individuals (data confidentiality)
- How: encrypt the original data; anyone can see the encrypted data, but only authorised individuals can decrypt to see the original data
- Used for both sending data across network and storing data on a computer system

Model of Encryption for Confidentiality



Model of Encryption for Confidentiality



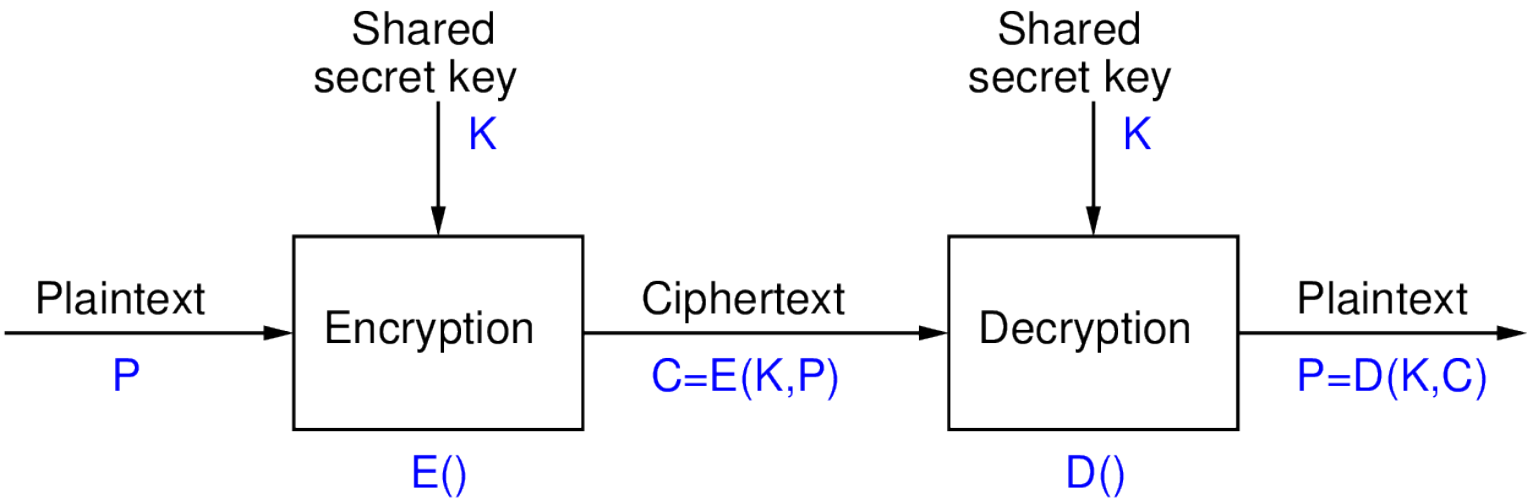
- Symmetric Cryptography: Key1 = Key 2
- Public Key Cryptography: Key1 $\neq$ Key 2

Terminology

Plaintext	original message
Ciphertext	encrypted or coded message
Encryption	convert from plaintext to ciphertext (enciphering)
Decryption	restore the plaintext from ciphertext (deciphering)
Key	information used in cipher known only to sender/receiver
Cipher	a particular algorithm (cryptographic system)
Cryptography	study of algorithms used for encryption
Cryptanalysis	study of techniques for decryption without knowledge of the key
Cryptology	cryptography + cryptanalysis

Contents

Symmetric Key Encryption for Confidentiality



Requirements

- Strong encryption algorithm: given algorithm, ciphertext or known pairs of (plaintext, ciphertext), attacker should not be able to find plaintext or key
- Shared secret keys: sender and receiver both have a shared secret key; no one else knows the key

Classification of Symmetric Ciphers

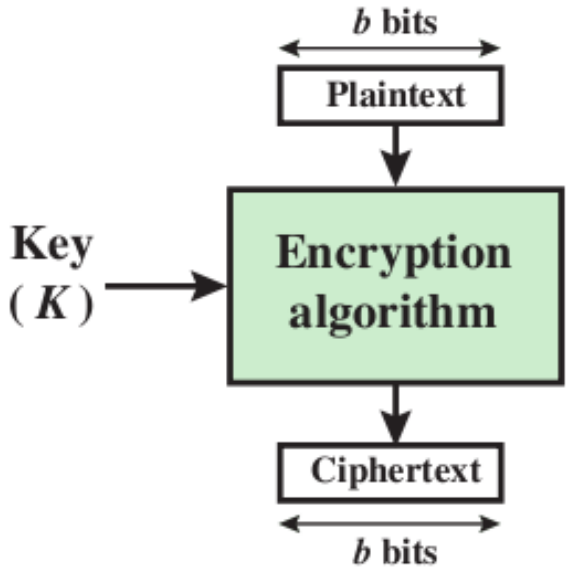
Processing of plaintext

Block ciphers process one block of elements at a time

Stream ciphers process input elements continuously

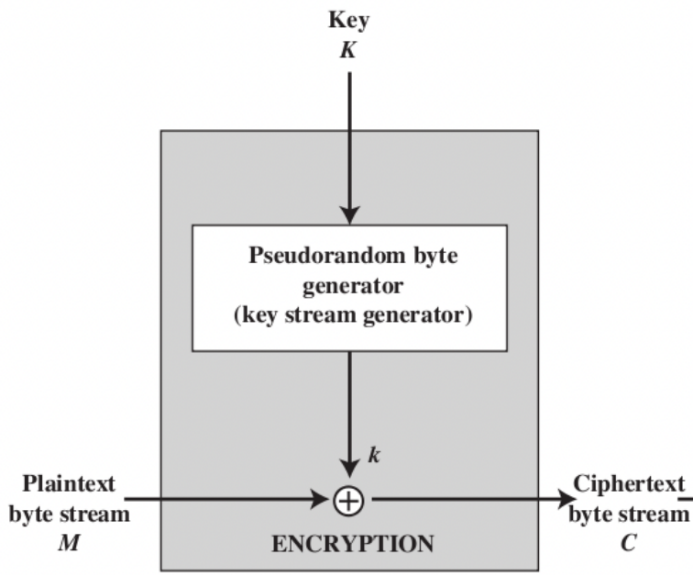
Block Ciphers

- Encrypt plaintext block by block, typically 64 or 128 bits
- Encryption performed by scrambling plaintext and key (permutes plaintext blocks)
- Different modes of operation with probabilistic encryption for improved security
- Widely used in e-commerce



Stream Ciphers

- Encrypt plaintext by bits/bytes/words
- Encryption performed by XOR plaintext with keystream (created by pseudo-random number generator)
- Fast algorithms/implementations in hardware
- Cannot reuse the keystream



Attacks

Goal of the Attacker

- Discover the plaintext (good)
- Discover the key (better)

Assumed Attacker Knowledge

- Ciphertext
- Algorithm
- Other pairs of (plaintext, ciphertext) using same key

Attack Methods

Brute-force attack Try every possible key on ciphertext

Cryptanalysis Exploit characteristics of algorithm to deduce plaintext or key

Assumption: attacker can recognise correct plaintext

Mono-alphabetic (Substitution) Ciphers

- Mono-alphabetic: use a single alphabet for both plaintext and ciphertext.

Mono-alphabetic (Substitution) Ciphers

- Mono-alphabetic: use a single alphabet for both plaintext and ciphertext. For instance

  Plain: ABCDE FGHIJ KLMNO PQRST UVWXY Z
  Cipher: CAOSY IBJTZ RDKUN FMVEG PWLHQ X

- Arbitrary substitution: one element maps to any other element
  - n element alphabet allows $n!$ permutations or keys
  - English: 26! permutations/keys
- Security
  - $n!$, e.g., there're 29! keys in Norwegian ($> 10^{30}$)
  - Vulnerable to frequency attacks

Attacks on Mono-alphabetic Ciphers

- Exploit the regularities of the language
  - Frequency of letters, digrams, trigrams
  - Expected words
- Fundamental problem with mono-alphabetic ciphers
  - Ciphertext reflects the frequency data of original plaintext
  - Solution 1: encrypt multiple letters of plaintext
  - Solution 2: use multiple cipher alphabets

Playfair Cipher - Initialization

1. Create 5x5 matrix and write keyword (row by row)
2. Fill out remainder with alphabet, not repeating any letters
3. **Special**: Treat I and J as same letter

Playfair Cipher - Encryption

1. Operate on pair of letters (digram) at a time
2. **Special**: if digram with same letters, separate by special letter (e.g. x)
3. Plaintext in same row: replace with letters to right
4. Plaintext in same column: replace with letters below
5. Else, replace by letter in same row as it and same column as other plaintext letter

- Plaintext: `helloworld`
- Keyword: `security`

S	E	C	U	R
I	T	Y	A	B
D	F	G	H	K
L	M	N	O	P
Q	V	W	X	Z

Plaintext: `HE LX LO WO RL DX`
Ciphertext: `FU OQ MP ... HQ`

Security

- large keyspace
- frequency attack: table of $25^2 = 625$ entries/digrams
- relatively easy (digrams, trigrams, expected words)

Vigenère Cipher

- Set of 26 general Caesar ciphers

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

- Letter in key determines the Caesar cipher to use
- Example: key = uib

  Plain: INTERNETTECHNOLOGIES
  Key: UIBUIBUIBUIBUIBUIBUIBUI
  Cipher: CVUYZOYBUYKIHWMIOJYA

- Multiple ciphertext letters for each plaintext letter, e.g., the first I→C and the second I→J

Vigenère Cipher - Is it Breakable?

- Yes
- Monoalphabetic or Vigenère cipher? Letter frequency analysis
- Determine length of keyword
- For keyword length m, Vigenère is m mono-alphabetic substitutions
- Break the mono-alphabetic ciphers separately
- Weakness is repeating, structured keyword

How to destroy the structure of keyword in the key string?

- combine the keyword and plaintext: Autokey cipher
- have sufficiently long key: Vernam cipher

Rotor Machine - Enigma

- Web-based Simulator of Enigma
- Web-based Simulator of Turing Bomb

Example Transposition Cipher: Rail Fence

Rail fence

Encryption: Plaintext letters written in diagonals over K rows; ciphertext obtained by reading row-by-row. For example,

- plaintext: attackpostponed
- key: 3
- ciphertext: ACSNTAKOTOETPPD

Column-Transposition Cipher

Encryption: Ciphertext obtained by reading column-by-column, but re-arranged transposition according to the key.
For example

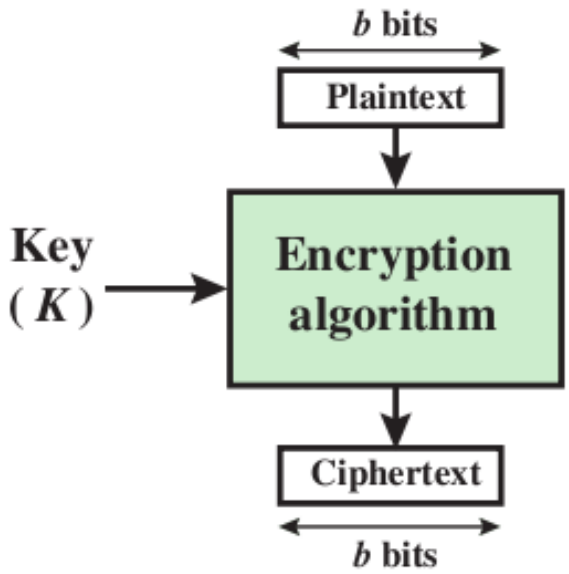
- plaintext: attackpostponeduntiltwoamxyz
- key: 4312567
- ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Product System

Combine substitution and transposition and repeat them n times. For example,

1. Apply Vigenere cipher with $K_{n,1}$
2. Apply Rail fence cipher with $K_{n,2}$

Modern Block Ciphers



- Encrypt a block of plaintext as a whole to produce same sized ciphertext
- Typical block sizes are 64 or 128 bits
- Modes of operation used to apply block ciphers to larger plaintexts
- Two widely used standard: DES and AES

Advanced Encryption Standard (AES)

- NIST held competition to select algorithm to replace DES/3DES in 1997
  - Won by Rijndael algorithm by Rijmen and Daemen
  - 2001: Standardised as FIPS-197
- Block size: 128
- Key length: 128, 192, 256 bits
- Substitution-permutation network
- Status: used in many products, e.g. WiFi (WPA), full disk encryption (BitLocker, FileVault2, dm-crypt, LUKS), Internet security (HTTPS), ...
- AES-NI Intel instruction

AES-Rijndael

Vincent Rijmen & Joan Daemen

Vincent is now an adjunct professor in the Selmer Center

Brute Force Attacks - Time Cost

- Age of Earth: 4×10^9 years
- Age of Universe: 1.3×10^{10} years

Brute Force Attacks - Time Cost

- Age of Earth: 4×10^9 years
- Age of Universe: 1.3×10^{10} years

Key length	Key space	Worst case time at speed: 10^9/sec	10^{12}/sec	10^{15}/sec
32	2^{32}	4 sec	4 ms	4 us
56	2^{56}	833 days	20 hrs	72 sec
64	2^{64}	584 yrs	213 days	5 hours
128	2^{128}	10^{22} yrs	10^{19} yrs	10^{16} yrs

Brute Force Attacks - Time Cost

- Age of Earth: 4×10^9 years
- Age of Universe: 1.3×10^{10} years

Key length	Key space	Worst case time at speed: 10^9/sec	10^{12}/sec	10^{15}/sec
32	2^{32}	4 sec	4 ms	4 us
56	2^{56}	833 days	20 hrs	72 sec
64	2^{64}	584 yrs	213 days	5 hours
128	2^{128}	10^{22} yrs	10^{19} yrs	10^{16} yrs
192	2^{192}	10^{41} yrs	10^{38} yrs	10^{35} yrs
256	2^{256}	10^{60} yrs	10^{57} yrs	10^{54} yrs

How Fast/Expensive is a Brute Force Attack Today?

```
OpenSSL> speed aes-128-cbc
Doing aes-128 cbc for 3s on 16 size blocks: 35923327 aes-128 cbc's in 2.98s
Doing aes-128 cbc for 3s on 64 size blocks: 9097135 aes-128 cbc's in 2.95s
Doing aes-128 cbc for 3s on 256 size blocks: 2340300 aes-128 cbc's in 2.99s
Doing aes-128 cbc for 3s on 1024 size blocks: 567862 aes-128 cbc's in 2.97s
Doing aes-128 cbc for 3s on 8192 size blocks: 68946 aes-128 cbc's in 2.94s
Doing aes-128 cbc for 3s on 16384 size blocks: 30720 aes-128 cbc's in 2.81s
OpenSSL 1.1.1d  10 Sep 2019
built on: Thu Sep 12 09:32:33 2019 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) idea(int) blowfish(ptr)
compiler: clang -fPIC -arch x86_64 -O3 -Wall -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -D
_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DVPAES_
 -DNDEBUG
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
aes-128 cbc      192876.92k   197361.57k   200373.51k   195788.11k   192110.76k   179116.19k
```

How Fast/Expensive is a Brute Force Attack Today?

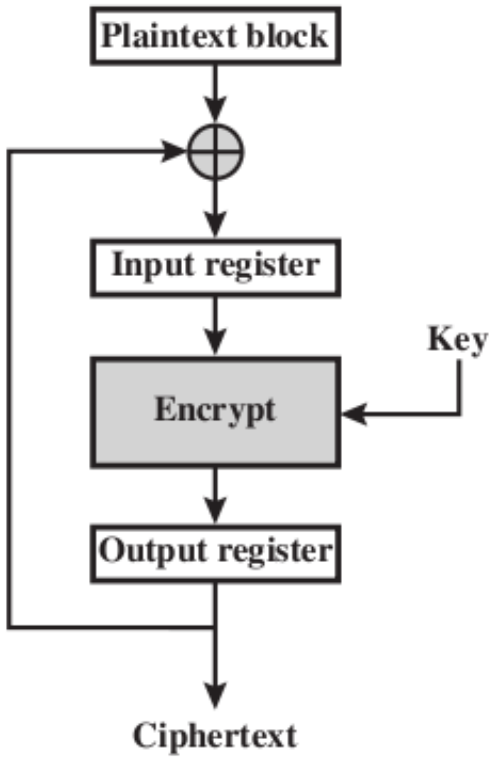
```
OpenSSL> speed aes-128-cbc
Doing aes-128 cbc for 3s on 16 size blocks: 35923327 aes-128 cbc's in 2.98s
Doing aes-128 cbc for 3s on 64 size blocks: 9097135 aes-128 cbc's in 2.95s
Doing aes-128 cbc for 3s on 256 size blocks: 2340300 aes-128 cbc's in 2.99s
Doing aes-128 cbc for 3s on 1024 size blocks: 567862 aes-128 cbc's in 2.97s
Doing aes-128 cbc for 3s on 8192 size blocks: 68946 aes-128 cbc's in 2.94s
Doing aes-128 cbc for 3s on 16384 size blocks: 30720 aes-128 cbc's in 2.81s
OpenSSL 1.1.1d  10 Sep 2019
built on: Thu Sep 12 09:32:33 2019 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) idea(int) blowfish(ptr)
compiler: clang -fPIC -arch x86_64 -O3 -Wall -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -D
_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DVPAES_
 -DNDEBUG
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
aes-128 cbc      192876.92k   197361.57k   200373.51k   195788.11k   192110.76k   179116.19k
```

- 128-bit key
- My MacPro: $\approx 2 \times 10^8$ blocks/sec
- Brute Force: $> 10^{22}$ years
- Age of Universe: 1.3×10^{10} years

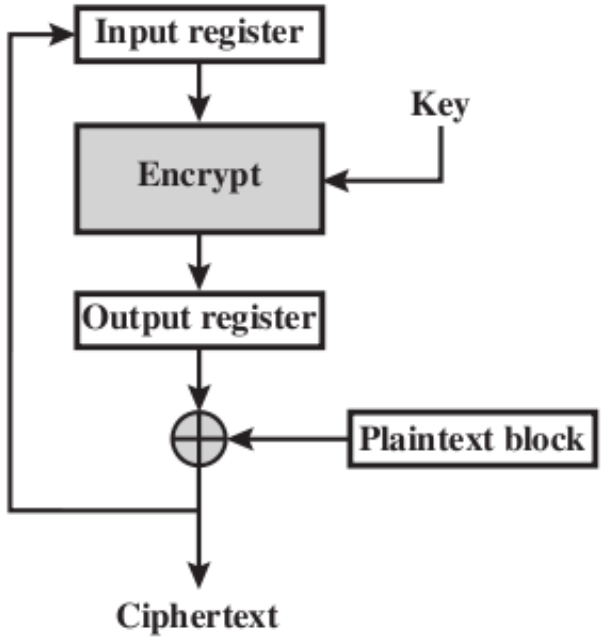
Using Block Ciphers on Real Data

- Block ciphers typically operate on 64 or 128 bit blocks
- Modes of operation are used to apply ciphers on multiple blocks
  - Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback Mode (CFB), Output Feedback Mode (OFB), Counter (CTR), XTS-AES
- Trade-offs: security, parallelism, error propagation
- Often require Initialisation Vector (IV)
- CFB, OFB and CTR can turn block cipher into stream cipher

Feedback: CBC and CFB

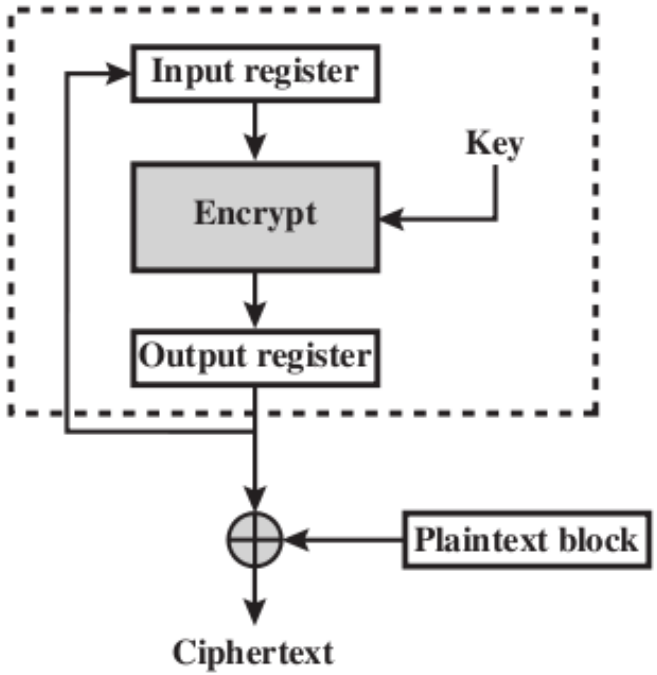


(a) Cipher block chaining (CBC) mode

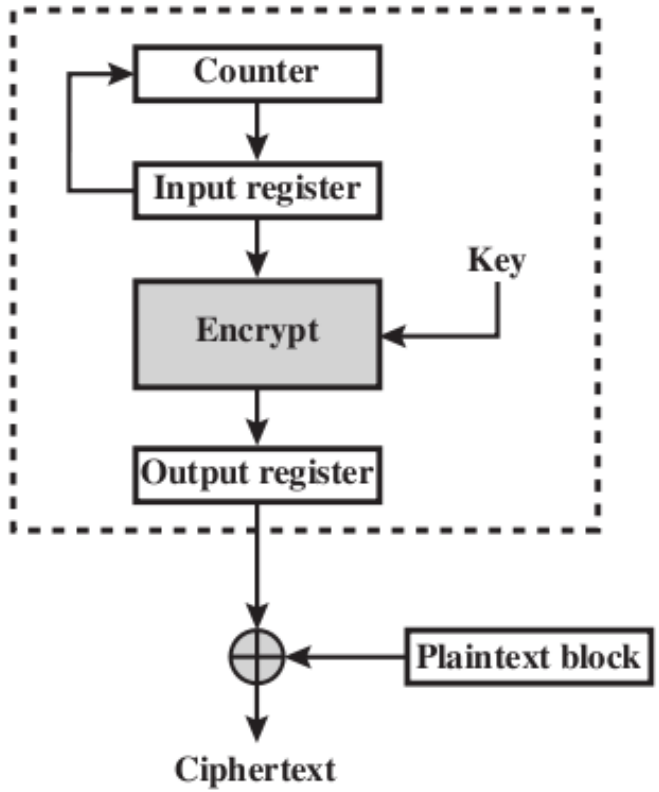


(b) Cipher feedback (CFB) mode

Feedback: OFB and CTR



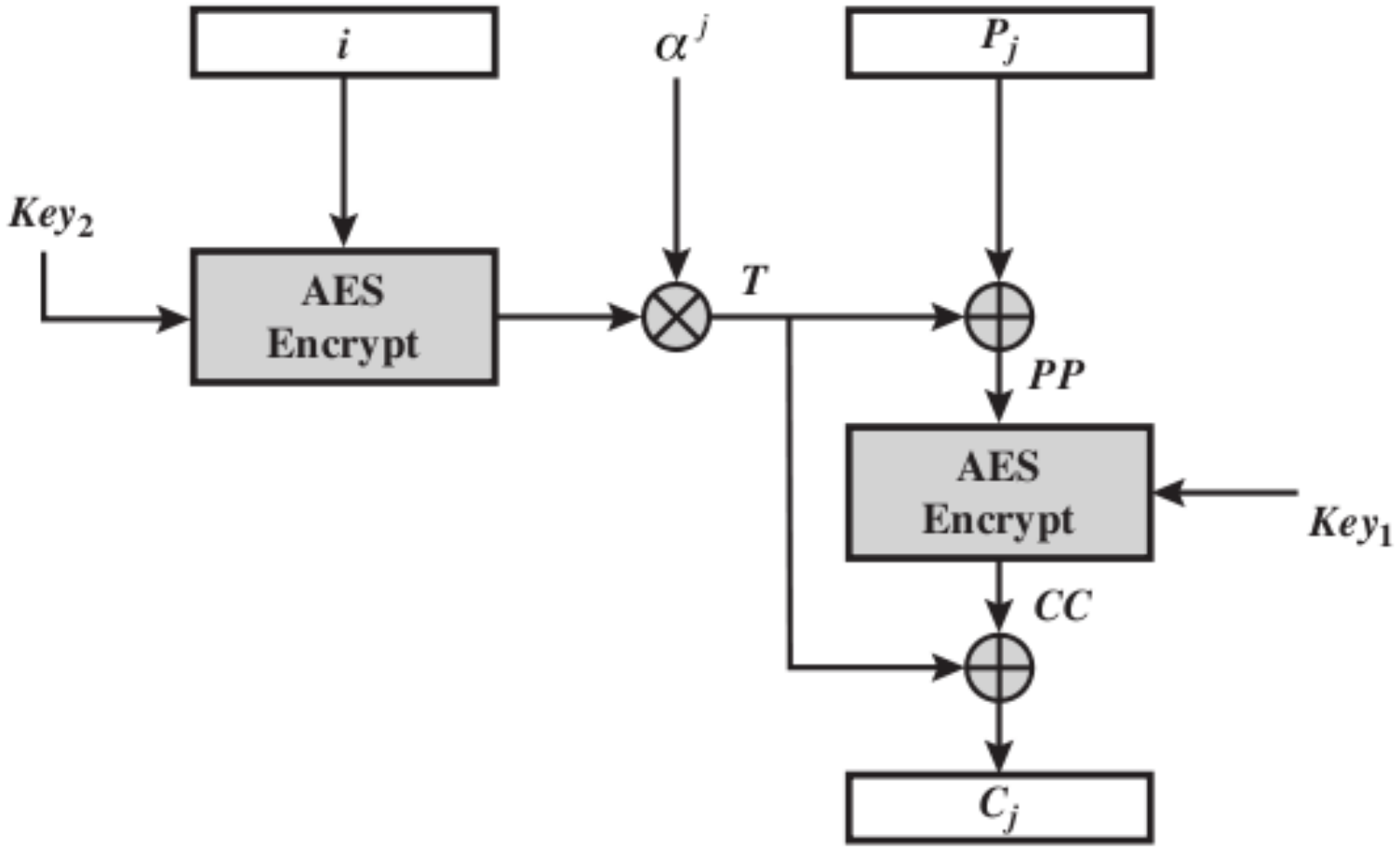
(c) Output feedback (OFB) mode



(d) Counter (CTR) mode

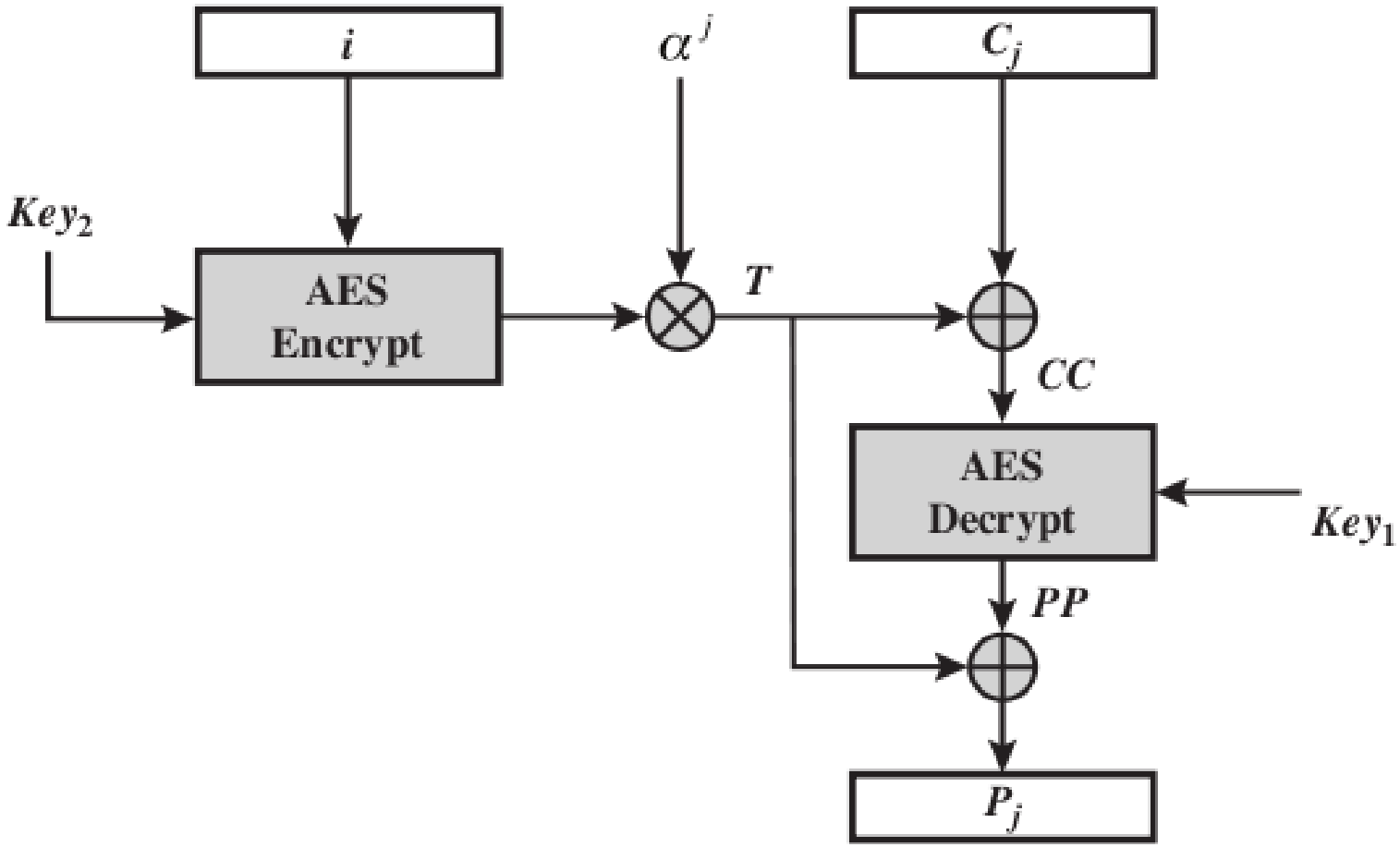
- Works like a keystream generator for a stream cipher.
- Keystream can be generated in advance or in parallel.
- Can be used to encrypt stored data.

XTS-AES Encryption of Single Block

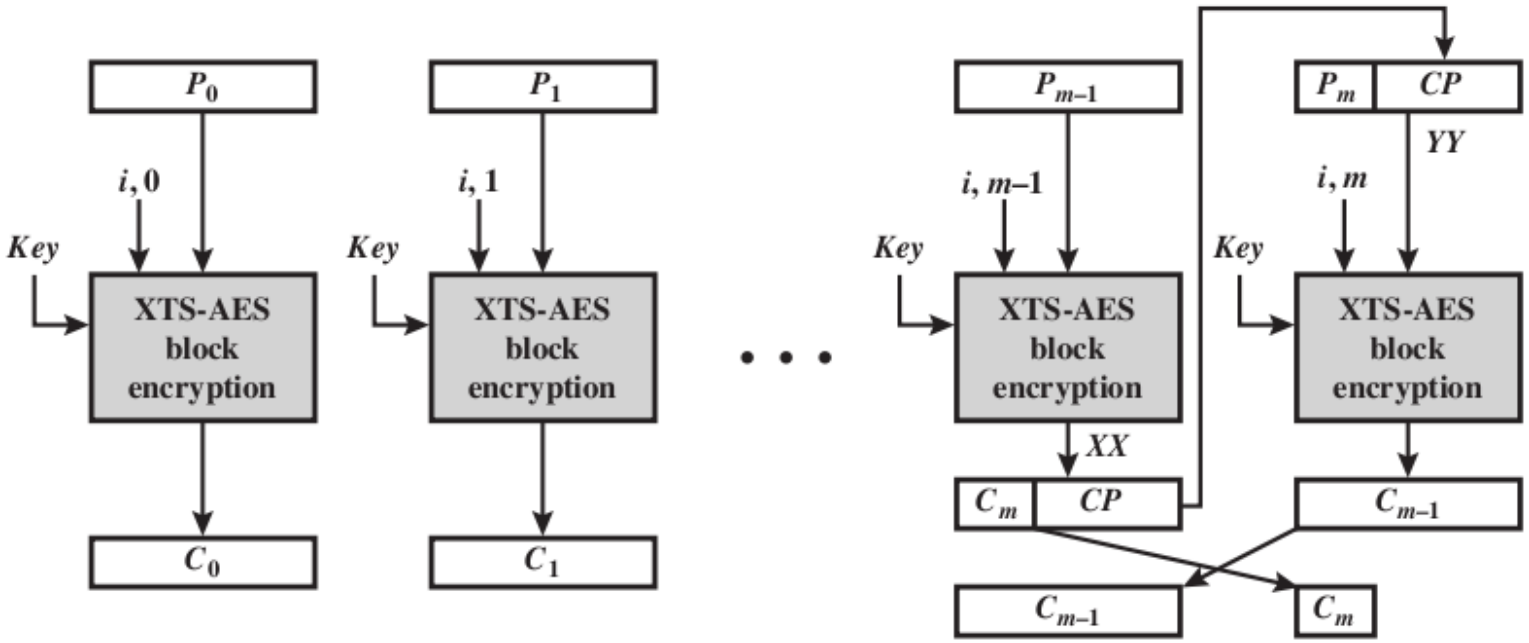


- i is the number of the sector
- j is the number of the block within the sector

XTS-AES Decryption of Single Block

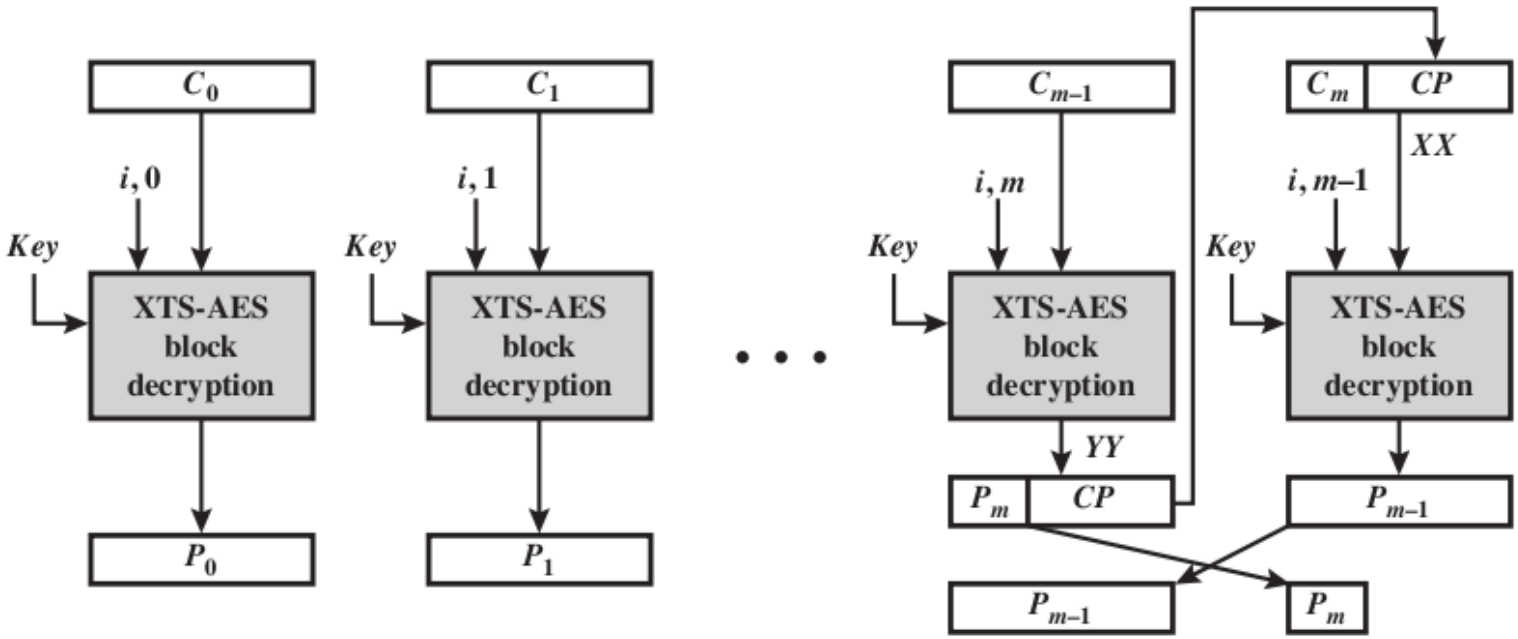


XTS-AES Encryption

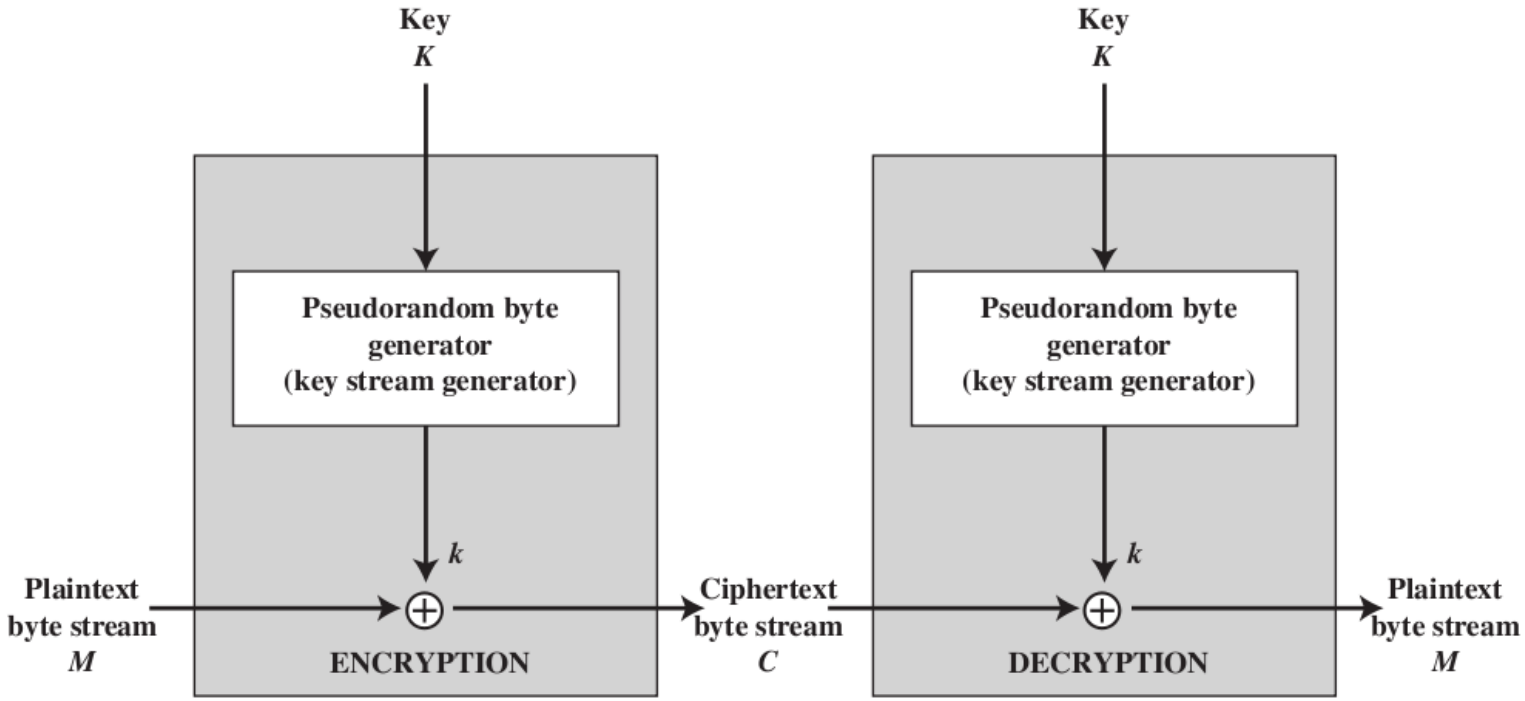


Designed for encrypting stored data (as opposed to transmitted data).

XTS-AES Decryption



Stream Ciphers



- Encrypt one byte at a time by XOR with pseudo-random byte (keystream)
- Generally faster implementations than block ciphers
- Keystream should not repeat (large period); use different key or nonce when reusing cipher

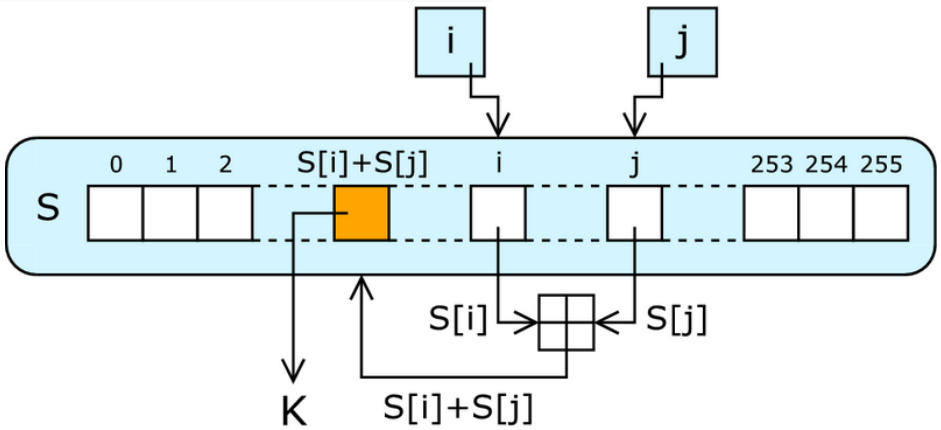
Example Stream Cipher: RC4

- Designed by Ron Rivest in 1987
- Used in secure web browsing (TLS before ver. 1.3) and wireless LANs
- Can use variable size key: 8 to 2048 bits
- Several theoretical limitations of RC4

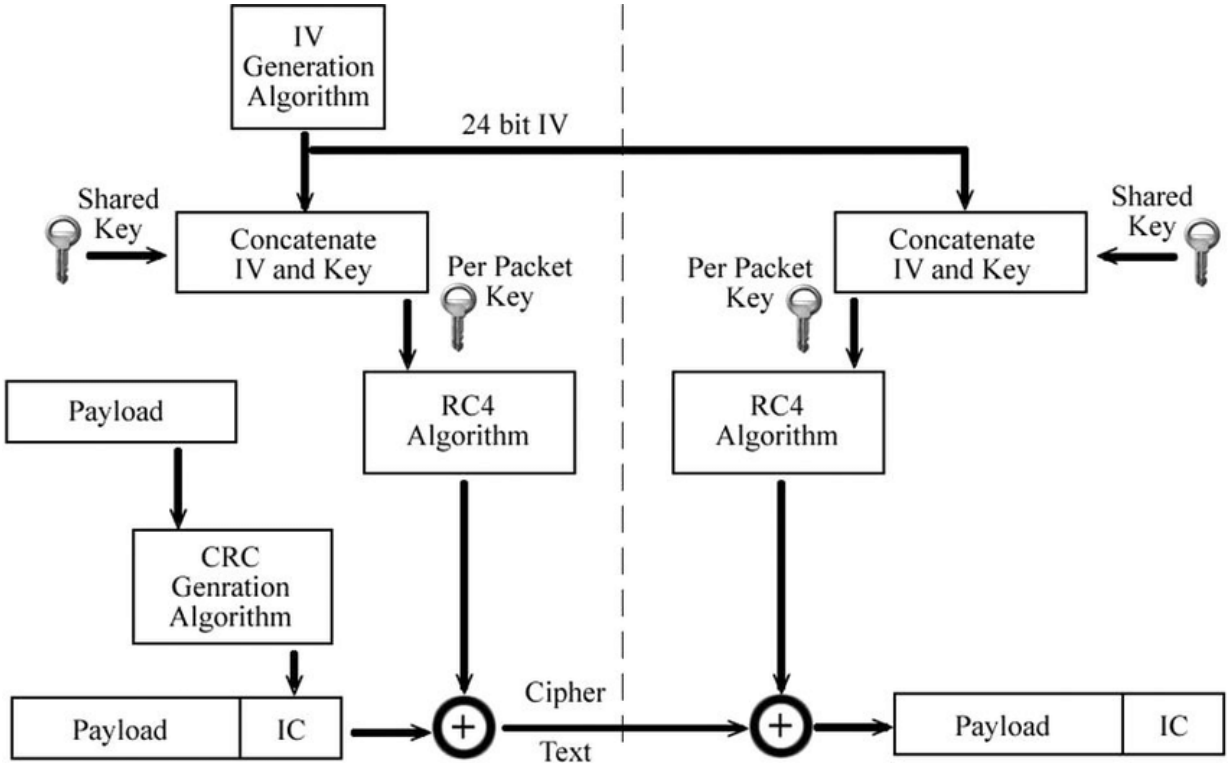
NB: RC4 is considered not secure enough, and was removed from modern security protocols like TLS 1.3

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap values of S[i] and S[j]
endfor
```

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```



Wired Equivalent Privacy (WEP) for WiFi



- IV is small, static and in cleartext (24-bit IV leads to collision with probability 50% after 5000 packets)
- IV is a part of encryption key (64/128 bits)

Contents

Authentication

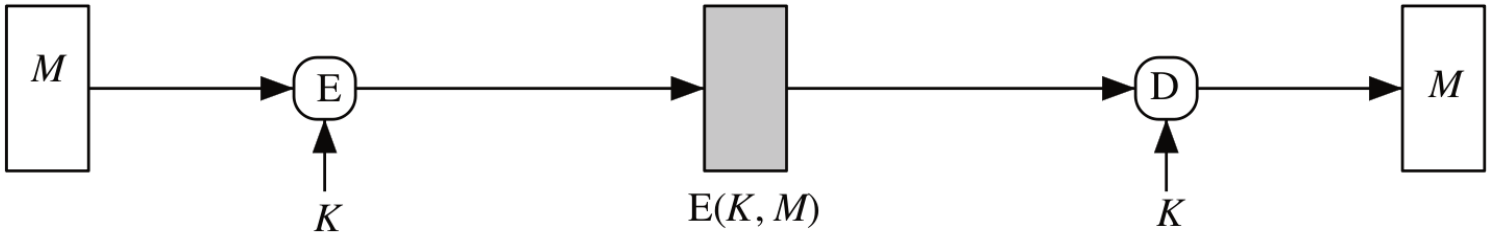
Integrity and authenticity usually come in pairs

Authentication=integrity+authenticity

- Authentication: receiver wants to verify
  1. Contents of the message have not been modified (*data integrity*)
  2. Source of message is as claimed (*source authenticity*)
- Different approaches available
  - Symmetric key encryption
  - **Hash functions**
  - **Message Authentication Codes (MAC)**
  - Digital signatures (public key cryptography)

Authentication Using Symmetric Key Encryption

- Assumption: decryption using wrong key or modified ciphertext will produce unintelligible output
  - but what if a stream cipher is used and attacker changes just one bit?
- Symmetric key encryption can provide: data authentication and source authentication (as well as confidentiality)



- However, typically authentication is performed separately from encryption for confidentiality
  - avoid overhead of using encryption when not needed

Hash Functions (Message Digest)

A cryptographic hash function, H(), takes

- a **variable size** input message M,
- and produces a **fixed size**, small output hash h, i.e.

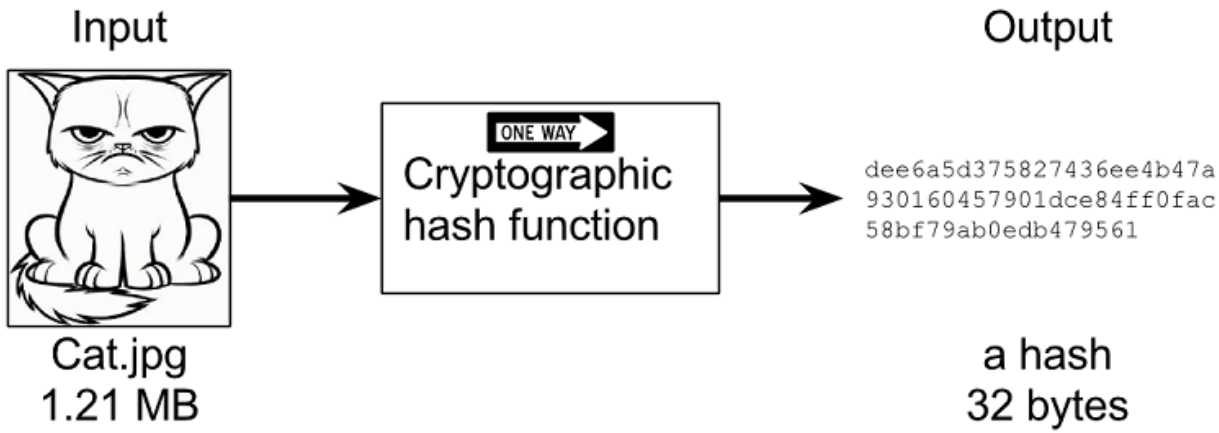
$$h = \mathrm{H}(M).$$

Hash Functions (Message Digest)

A cryptographic hash function, H(), takes

- a **variable size** input message M,
- and produces a **fixed size**, small output hash h, i.e.

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



Security Requirements

Preimage resistant: for any given h, computationally infeasible to find y such that $\mathrm{H}(y) = h$ (*one-way property*)

Second preimage resistant: for any given x, computationally infeasible to find $y \neq x$ with $\mathrm{H}(y) = \mathrm{H}(x)$ (*weak collision resistant*)

Collision resistant: computationally infeasible to find any pair (x, y) such that $\mathrm{H}(x) = \mathrm{H}(y)$ (*strong collision resistant*)

Security Requirements

Preimage resistant: for any given h, computationally infeasible to find y such that $\mathrm{H}(y) = h$ (*one-way property*)

Second preimage resistant: for any given x, computationally infeasible to find $y \neq x$ with $\mathrm{H}(y) = \mathrm{H}(x)$ (*weak collision resistant*)

Collision resistant: computationally infeasible to find any pair (x, y) such that $\mathrm{H}(x) = \mathrm{H}(y)$ (*strong collision resistant*)

Brute Force Attacks

The complexity depends on hash value length of n bits

- Preimage and second preimage resistant: 2^n
- Collision resistant: $2^{n/2}$ (birthday paradox)

Authentication Using Hash Functions

- Hash function H: variable-length input data M; fixed-size output hash value $h = \mathrm{H}(M)$
- Applying H to large set of inputs should produce evenly distributed and random looking outputs
- Cryptographic hash function: computationally infeasible to find
  1. M that maps to known h (one-way property)
  2. M_1 and M_2 that produce same h (collision resistance property)
- Send hash value over **secure channel**; receiver verifies if message changed

Required Properties when Using Hash Functions

Not all applications of hash functions require all properties

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Credit: Table 11.2 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

Hash Algorithms: MD5

- Message Digest algorithm 5, developed by Ron Rivest in 1991
- Standardised by IETF in RFC 1321
- Generates 128-bit hash
- Was commonly used by applications, passwords, file integrity; no longer recommended
- Collision and other attacks possible, tools publicly available

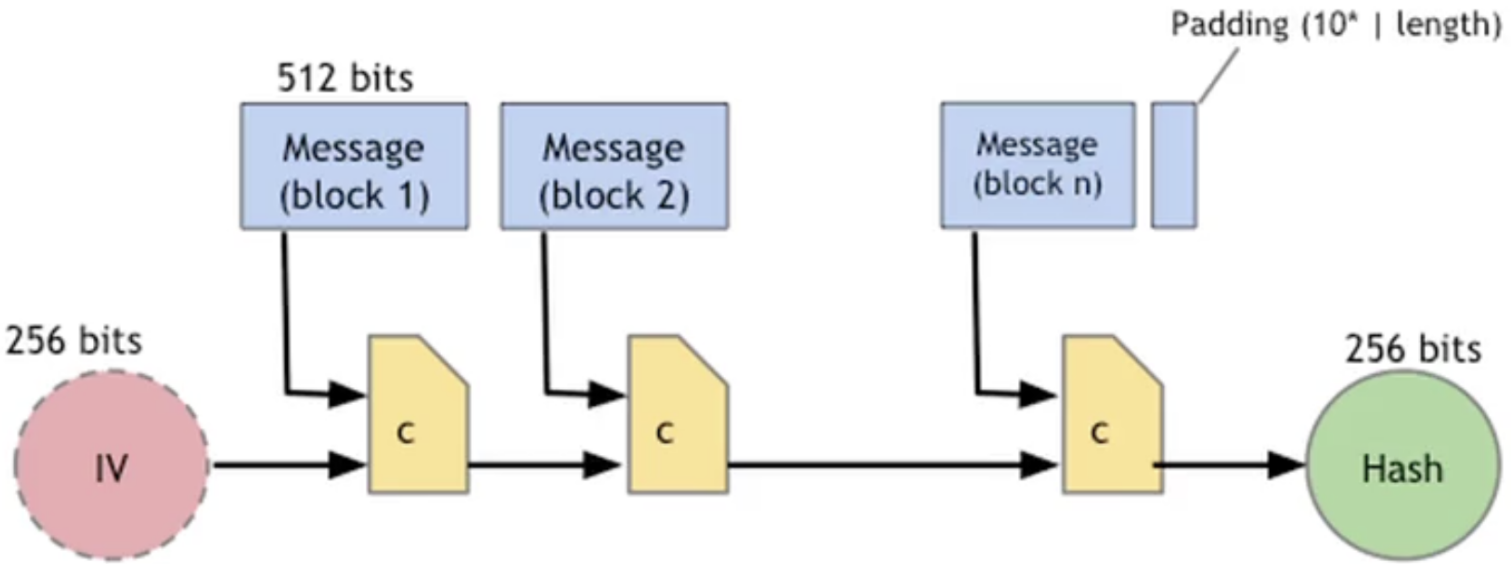
Secure Hash Algorithms: SHA series

- Published by NIST in FIPS 180 series from 1993
- Developed over time: SHA-0, SHA-1, SHA-2, SHA-3
- SHA-0 and SHA-1 are insecure, no longer recommended
- SHA-2 considered secure (SHA-224, etc. versions)
- SHA-3 is standardized in 2015 and deployed (e.g., by Ethereum)

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Credit: Table 11.3 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

General Structure of SHA-256

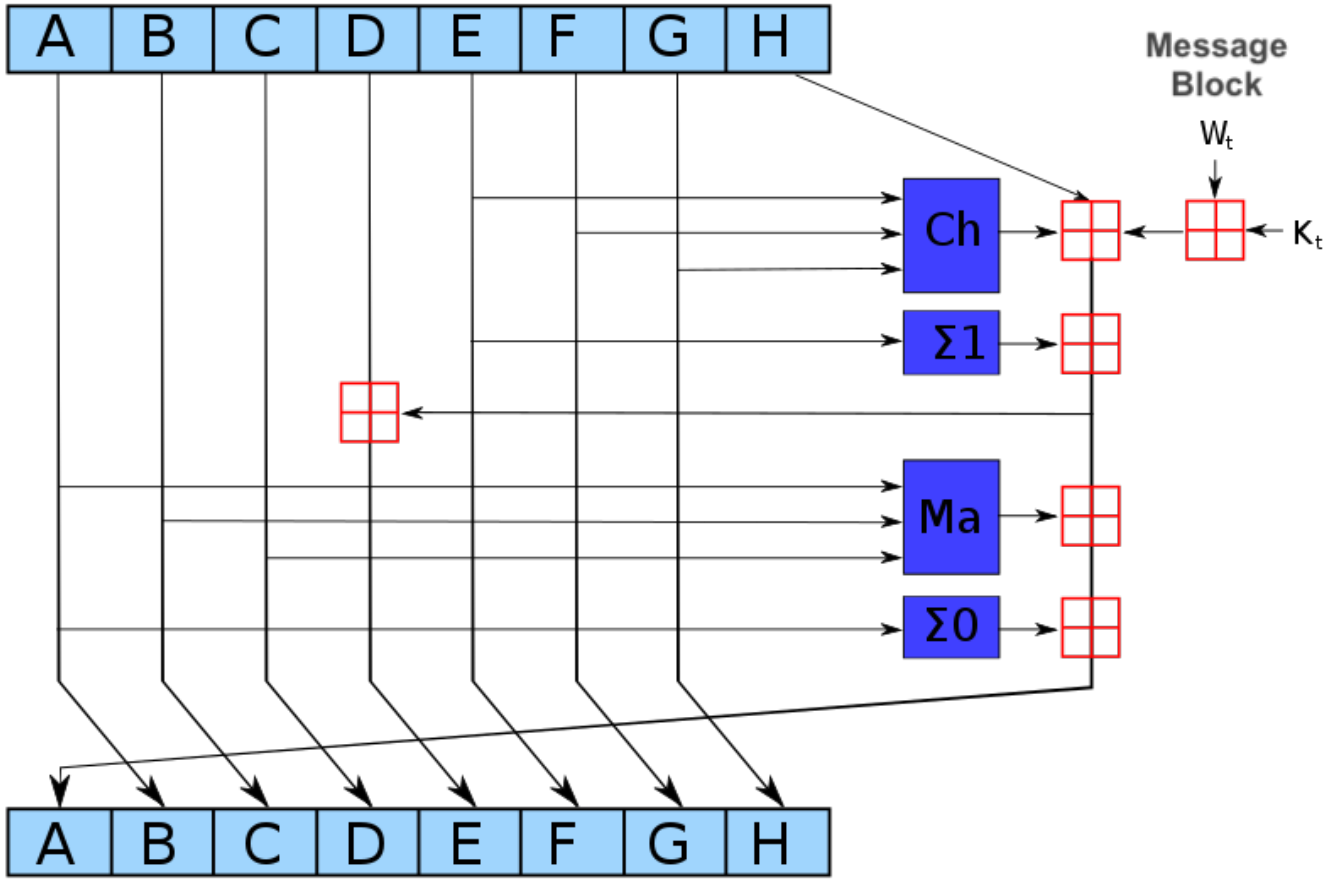


- An input M is padded such that the length is a multiple of 512 bits long
- M is parsed into message blocks $M_1, M_2, \cdots, M_N$
- The messages blocks are processed one at a time:

$$H_i = H_{i-1} \boxplus C_{M_i}(H_{i-1})$$

where $\boxplus$ is word-wise mod 2^{32} addition and C is a compression function with 256-bit output

SHA-256 compression function C (64 iterations)



- $Ch(E, F, G) = (E \wedge F) \oplus (\overline{E} \wedge G)$
- $Ma(A, B, C) = (A \wedge B) \oplus (B \wedge C) \oplus (A \wedge C)$
- $\sum_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$
- $\sum_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$

Overview of SHA Family

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Rounds	Operations	Security (in bits) against collision attacks
MD5 (as reference)		128	128 (4 × 32)	512	64	And, Xor, Rot, Add (mod 2^{32}), Or	≤18 (collisions found)[58]
SHA-0		160	160 (5 × 32)	512	80	And, Xor, Rot, Add (mod 2^{32}), Or	<34 (collisions found)
SHA-1							<63 (collisions found)[59]
SHA-2	*SHA-224*	224	256 (8 × 32)	512	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112
	SHA-256	256					128
	SHA-384	384	512 (8 × 64)	1024	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192
	SHA-512	512					256
	SHA-512/224	224					112
	SHA-512/256	256					128
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	24[60]	And, Xor, Rot, Not	112
	SHA3-256	256		1088			128
	SHA3-384	384		832			192
	SHA3-512	512		576			256
	SHAKE128	*d* (arbitrary)		1344			min(*d*/2, 128)
	SHAKE256	*d* (arbitrary)		1088			min(*d*/2, 256)

Demo: MD5 and SHA-1 collisions

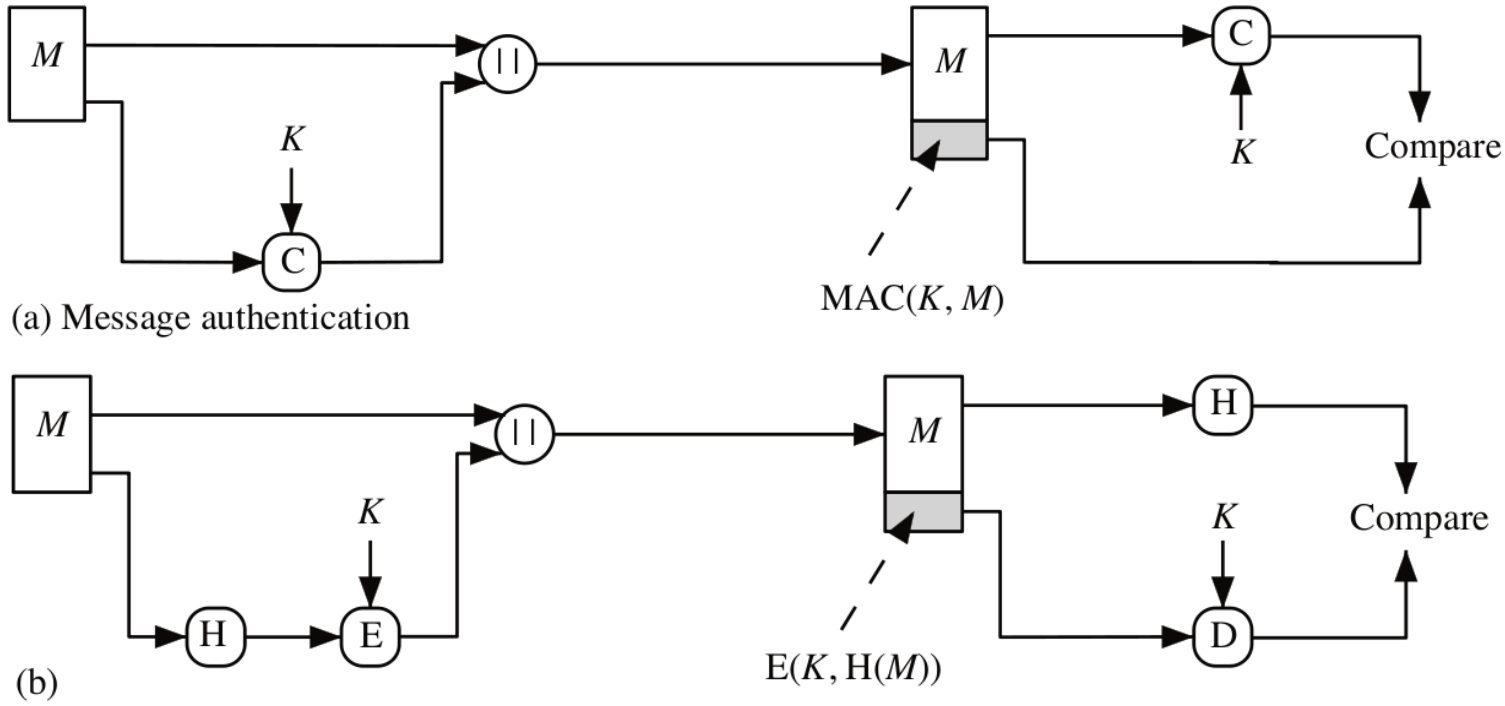
Message Authentication Codes (MAC)

- Append small, fixed-size block of data to message: cryptographic checksum or MAC

$$\mathrm{MAC} = \mathrm{F}(K, M)$$

  - M = input message
  - F = MAC function
  - K = shared secret key of k bits
  - MAC = message authentication code (or tag) of n bits
- MAC function also called *keyed hash function*
- MAC function similar to encryption, but does not need to be reversible
  - easier to design stronger MAC functions than encryption functions

Authentication Using Hash Function and MAC



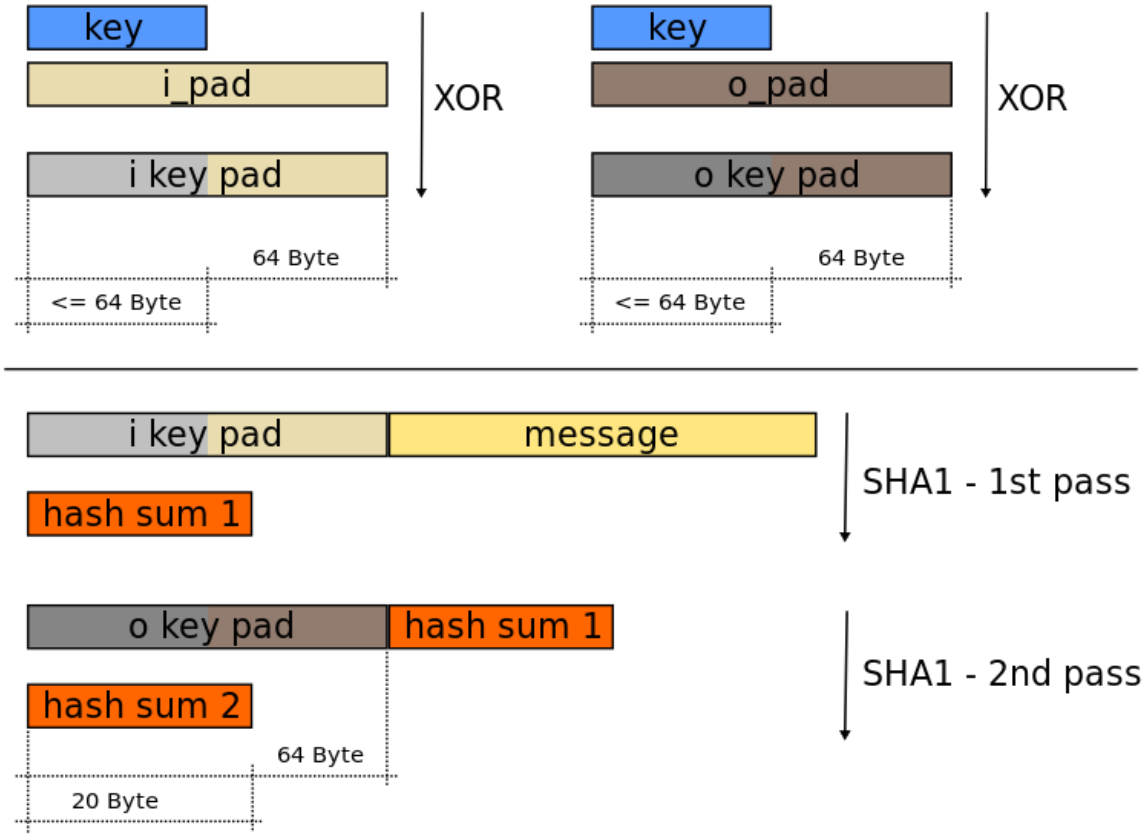
Credit: Figure from Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

MAC Construction

- Data Authentication Algorithm (DAA): based on DES; considered insecure
- Cipher-Based Message Authentication Code (CMAC): mode of operation used with Triple-DES and AES
- OMAC, PMAC, UMAC, VMAC, ...
- **HMAC**: MAC function derived from cryptographic hash functions
  - MD5/SHA are faster in software (compared to block ciphers)
  - Libraries for hash functions widely available
  - Security of HMAC depends on security of hash function used

HMAC

$$HMAC(K,m)=H((K\oplus opad)||H((K\oplus ipad)||m))$$



MAC Attacks

Security Requirements

- Key is secret and difficult to find from pairs of (M, MAC)
- Given pairs of (M, MAC), difficult to find the MAC of another message

Brute Force Attacks on MACs

- Option 1: Try all possible keys for one or more pairs of (M, MAC); effort $\approx 2^k$
- Option 2: Try many values of M to find a second preimage; effort $\approx 2^n$
- Effort to break MAC: $\min(2^k, 2^n)$

Assumptions: Authentication with Symmetric Key and MACs

- Entity receiving ciphertext that **successfully decrypts** with symmetric secret key K_{AB} knows that the original message has not been modified and that it originated at one of the owners of the secret key (i.e. A or B).
- Entity receiving a message with attached MAC that successfully verifies, knows that the message has not been modified and originated at one of the owners of the MAC secret key.

Summary on Symmetric Cryptographic Primitives

	Reversible	Key	Input/Output Size	Instances
Block Ciphers	Yes	Yes	Block Size	3DES, AES
Stream Ciphers	Yes	Yes	Byte or Word	RC4, Trivium
MACs	No	Yes	Variable/Fixed	CMAC, HMAC
Hash	No	No	Variable/Fixed	SHA-1/2/3

Common Features:

- iterative operations on the input (and key if in place)
- the output is a random-like binary string
  - this feature enables the design of cryptographic primitive based on other secure primitives

v • t • e	Block ciphers (security summary)
Common algorithms	AES · Blowfish · DES (internal mechanics, Triple DES) · Serpent · Twofish
Less common algorithms	ARIA · Camellia · CAST-128 · GOST · IDEA · LEA · RC2 · RC5 · RC6 · SEED · Skipjack · TEA · XTEA
Other algorithms	3-Way · Akelarre · Anubis · BaseKing · BassOmatic · BATON · BEAR and LION · CAST-256 · Chiasmus · CIKS-1 · CIPHERUNICORN-A · CIPHERUNICORN-E · CLEFIA · CMEA · Cobra · COCONUT98 · Crab · Cryptomeria/C2 · CRYPTON · CS-Cipher · DEAL · DES-X · DFC · E2 · FEAL · FEA-M · FROG · G-DES · Grand Cru · Hasty Pudding cipher · Hierocrypt · ICE · IDEA NXT · Intel Cascade Cipher · Iraqi · Kalyna · KASUMI · KeeLoq · KHAZAD · Khufu and Khafre · KN-Cipher · Kuznyechik · Ladder-DES · Libelle · LOKI (97, 89/91) · Lucifer · M6 · M8 · MacGuffin · Madryga · MAGENTA · MARS · Mercy · MESH · MISTY1 · MMB · MULTI2 · MultiSwap · New Data Seal · NewDES · Nimbus · NOEKEON · NUSH · PRESENT · Prince · Q · RC6 · REDOC · Red Pike · S-1 · SAFER · SAVILLE · SC2000 · SHACAL · SHARK · Simon · SM4 · Speck · Spectr-H64 · Square · SXAL/MBAL · Threefish · Treyfer · UES · xmx · XXTEA · Zodiac
Design	Feistel network · Key schedule · Lai–Massey scheme · Product cipher · S-box · P-box · SPN · Confusion and diffusion · Avalanche effect · Block size · Key size · Key whitening (Whitening transformation)
Attack (cryptanalysis)	Brute-force (EFF DES cracker) · MITM (Biclique attack · 3-subset MITM attack) · Linear (Piling-up lemma) · Differential (Impossible · Truncated · Higher-order) · Differential-linear · Distinguishing (Known-key) · Integral/Square · Boomerang · Mod *n* · Related-key · Slide · Rotational · Side-channel (Timing · Power-monitoring · Electromagnetic · Acoustic · Differential-fault) · XSL · Interpolation · Partitioning · Rubber-hose · Black-bag · Davies · Rebound · Weak key · Tau · Chi-square · Time/memory/data tradeoff
Standardization	AES process · CRYPTREC · NESSIE
Utilization	Initialization vector · Mode of operation · Padding

v • t • e	Stream ciphers	
Widely used ciphers	RC4 · block ciphers in stream mode · ChaCha	
eSTREAM Portfolio	**Software**	HC-256 · Rabbit · Salsa20 · SOSEMANUK
	Hardware	Grain · MICKEY · Trivium
Other ciphers	A5/1 · A5/2 · Achterbahn · E0 · F-FCSR · FISH · ISAAC · MUGI · ORYX · Panama · Phelix · Pike · Py · QUAD · Scream · SEAL · SNOW · SOBER · SOBER-128 · VEST · VMPC · WAKE	
Theory	shift register · LFSR · NLFSR · shrinking generator · T-function · IV	
Attacks	correlation attack · correlation immunity · stream cipher attacks	

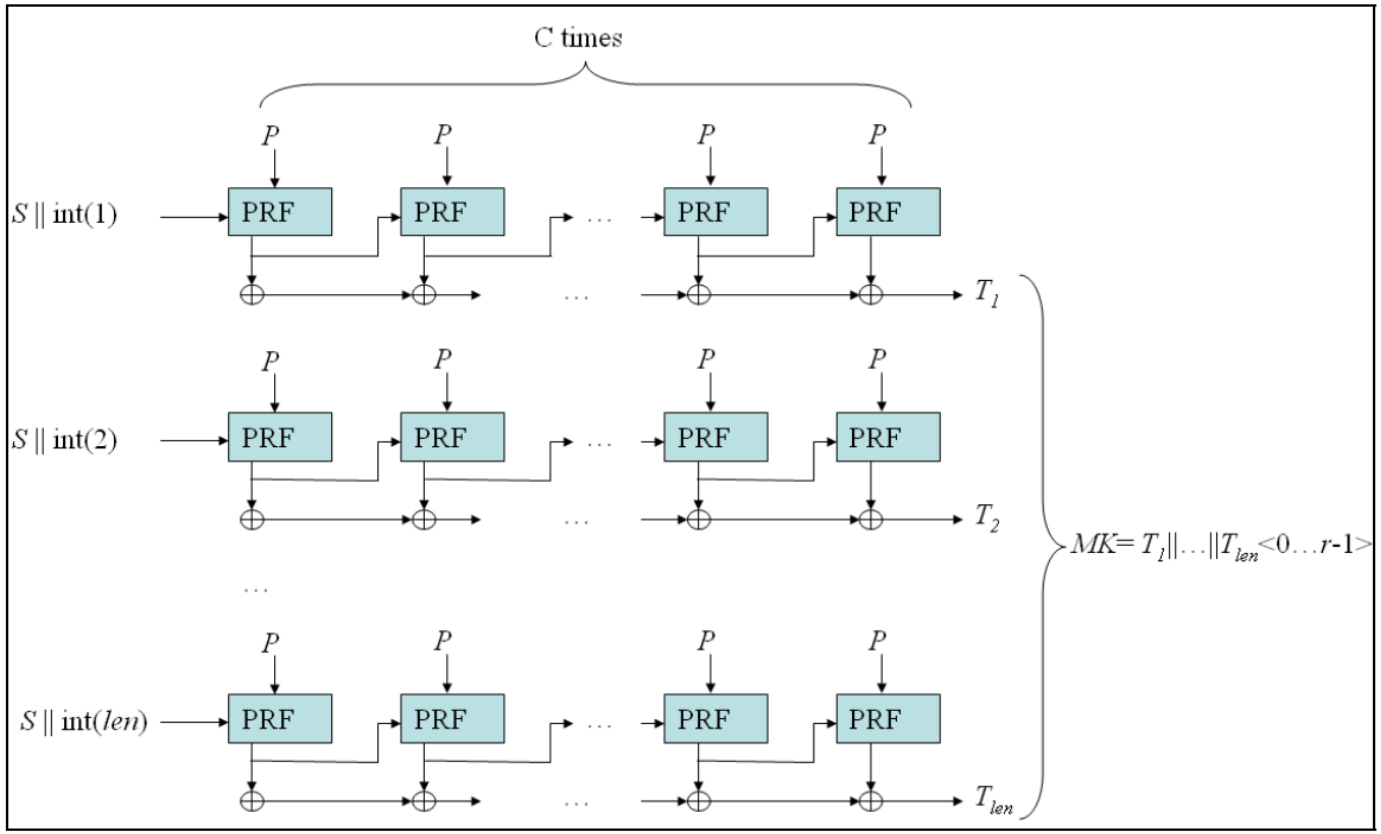
v·t·e	Cryptographic hash functions & message authentication codes
	List · Comparison · Known attacks
Common functions	MD5 · SHA-1 · SHA-2 · SHA-3 · BLAKE2
SHA-3 finalists	BLAKE · Grøstl · JH · Skein · Keccak (winner)
Other functions	BLAKE3 · CubeHash · ECOH · FSB · GOST · HAS-160 · HAVAL · Kupyna · MD2 · MD4 · MD6 · MDC-2 · N-Hash · RIPEMD · RadioGatún · SM3 · SWIFFT · Snefru · Streebog · Tiger · VSH · Whirlpool
Password hashing/ key stretching functions	Argon2 · Balloon · bcrypt · Catena · crypt · LM hash · Lyra2 · Makwa · PBKDF2 · scrypt · yescrypt
General purpose key derivation functions	HKDF · KDF2
MAC functions	DAA · CBC-MAC · GMAC · HMAC · NMAC · OMAC/CMAC · PMAC · VMAC · UMAC · Poly1305 · SipHash
Authenticated encryption modes	CCM · CWC · EAX · GCM · IAPM · OCB
Attacks	Collision attack · Preimage attack · Birthday attack · Brute-force attack · Rainbow table · Side-channel attack · Length extension attack
Design	Avalanche effect · Hash collision · Merkle–Damgård construction · Sponge function · HAIFA construction
Standardization	CRYPTREC · NESSIE · NIST hash function competition
Utilization	Hash-based cryptography · Merkle tree · Message authentication · Proof of work · Salt · Pepper
v·t·e	**Cryptography** [show]

Generation of Secret Keys

The security of symmetric cryptography depends on

- symmetric primitives: strength/quality of the design, implementation, etc.
- secret keys: key generation, key distribution, key management in general

Password-Based Key Derivation Func. (PBKDF)



MK = PBKDF2(P, S, C, dkLen)

- P: Password; S: Salt; C: number of iterations
- dkLen: intended length of derived key ($< 2^{32}$)
- len := dkLen/hLen
- int(i):=32-bit encoding of the integer i

$$len = Ceil(dkLen/hLen)$$