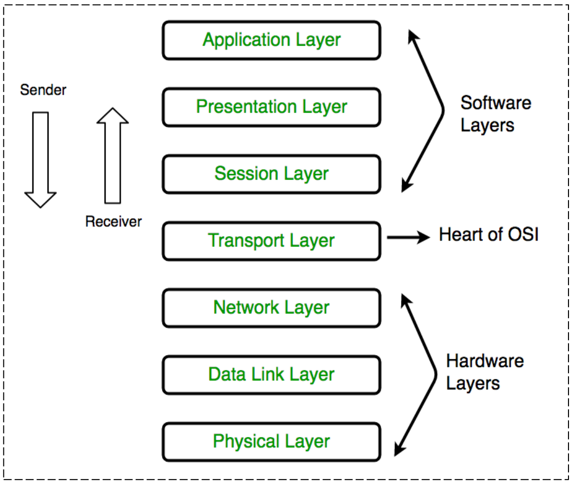
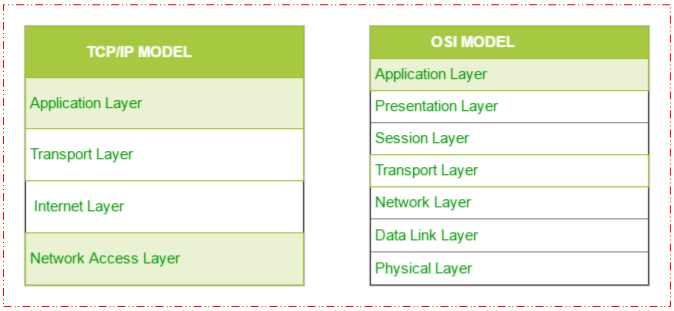
Cryptography Techniques

Professor DR. Bashar Al-Esawi

2022

Cryptography Techniques





TCP/IP MODEL
Application Layer
Transport Layer
Internet Layer
Network Access Layer

OSI MODEL
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

See also:
https://www.geeksforgeeks.org/difference-between-block-cipher-and-stream-cipher/?ref=lbp

Types Of Cryptography:

In general there are three types Of cryptography:

1.Symmetric Key Cryptography:

It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. The most popular symmetric key cryptography system is Data Encryption System(DES).

2.Hash Functions:

There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.

3.Asymmetric Key Cryptography:

Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.

Both **Substitution cipher technique** and **Transposition cipher technique** are the types of Traditional cipher which are used to convert the plain text into cipher text.

Substitution Cipher Technique:

In Substitution Cipher Technique plain text characters are replaced with other characters, numbers and symbols as well as in substitution Cipher Technique, character’s identity is changed while its position remains unchanged.

Transposition Cipher Technique:

Transposition Cipher Technique rearranges the position of the plain text’s characters. In transposition Cipher Technique, The position of the character is changed but character’s identity is not changed.

Block Cipher and **Stream Cipher** belongs to the symmetric key cipher. These two block ciphers and stream cipher are the methods used for converting the plain text into ciphertext.

The main difference between a **Block cipher** and a **Stream cipher** is that a block cipher converts the plain text into cipher text by taking plain text’s block at a time. While stream cipher Converts the plain text into cipher text by taking 1 byte of plain text at a time.

1. Monoalphabetic Cipher :

A monoalphabetic cipher is any cipher in which the letters of the plain text are mapped to cipher text letters based on a single alphabetic key. Examples of monoalphabetic ciphers would include the Caesar-shift cipher, where each letter is shifted based on a numeric key, and the atbash cipher, where each letter is mapped to the letter symmetric to it about the center of the alphabet.

2. Polyalphabetic Cipher :

A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The Vigenère cipher is probably the best-known example of a polyalphabetic cipher, though it is a simplified special case.

Difference between Confusion and Diffusion

Confusion and **diffusion** area unit the properties for creating a secure cipher. Each Confusion and diffusion area unit wont to stop the secret writing key from its **deduction or ultimately** for preventing the first message.

- ***Confusion is employed for making uninformed cipher text.***
- ***Diffusion is employed for increasing the redundancy of the plain text***.
- The stream cipher solely depends on Confusion,
- Diffusion is employed by each stream and block cipher.

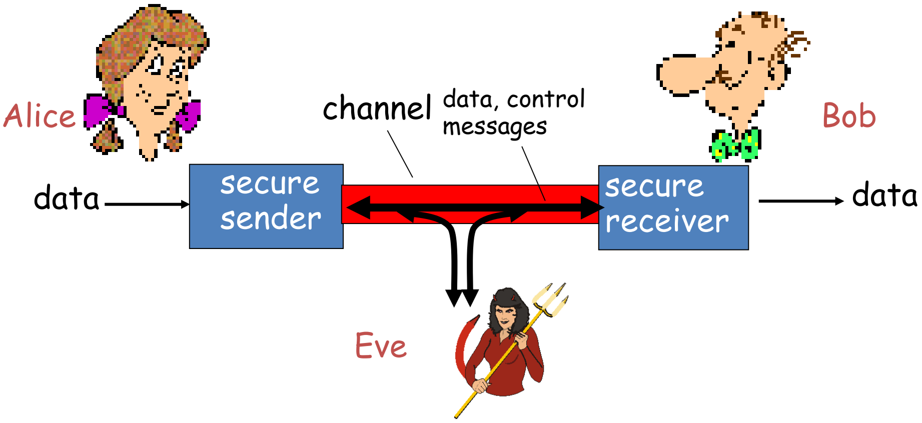
```
Confusion = Substitution
a --> b
Caesar Cipher
Diffusion = Transposition or Permutation
abcd --> dacb
DES
```

Confusion *is an encryption operation where the relationship between key and ciphertext is obscured.*
Diffusion *is an encryption operation where the influence of one plaintext symbol is spread over many ciphertext symbols with the goal of hiding statistical properties of the plaintext.*

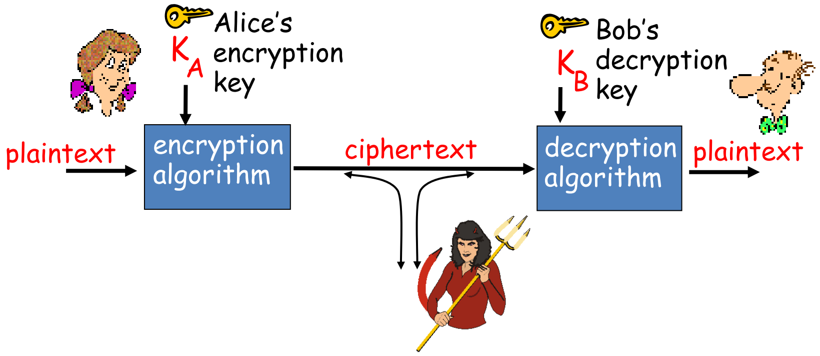
Confusion	Diffusion
Confusion protect the relationship between the ciphertext and key.	Diffusion protect the relationship between the ciphertext and plaintext.
If an individual bit in the key is changed, some bits in the ciphertext will also be modified.	If an individual symbol in the plaintext is changed, there are some symbols in the ciphertext will also be changed.
In confusion, the connection between the data of the ciphertext and the value of the encryption is made difficult. It is completed by substitution.	In diffusion, the numerical mechanism of the plaintext is used up into global statistics of the cipher text. This is achieved by permutation.
In confusion, vagueness is enhanced in resultant.	While in diffusion, redundancy is enhanced in resultant.
The relation among the cipher text and the key is concealed by confusion.	The relation among the cipher text and the plain text is concealed by diffusion.

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Eve (or Trudy, intruder) may intercept, delete, add messages



The language of cryptography

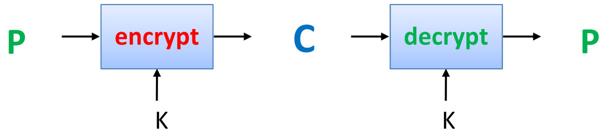


symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

Symmetric Cryptosystem

- **Scenario**
  - Alice wants to send a message (plaintext P) to Bob.
  - The communication channel is insecure and can be eavesdropped
  - If Alice and Bob have previously agreed on a symmetric encryption scheme and a secret key K, the message can be sent encrypted (ciphertext C)
- **Issues**
  - What is a good symmetric encryption scheme?
  - What is the complexity of encrypting/decrypting?
  - What is the size of the ciphertext, relative to the plaintext?

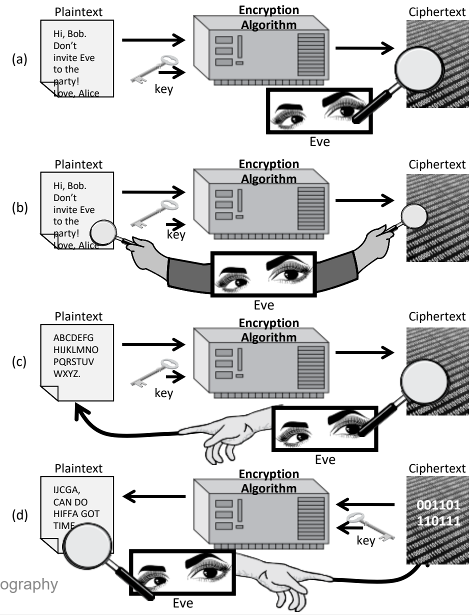


Basics

- **Notation**
  - Secret key K
  - Encryption function $E_K(P)$
  - Decryption function $D_K(C)$
  - Plaintext length typically the same as ciphertext length
  - Encryption and decryption are **one-one mapping functions** on the set of all n-bit arrays
- **Efficiency**
  - functions E_K and D_K should have efficient algorithms
- **Consistency**
  - Decrypting the ciphertext yields the plaintext
  - $\mathbf{D_K(E_K(P)) = P}$

Attacks

- **Attacker may have**
  a) collection of ciphertexts (**ciphertext only attack**)
  b) collection of plaintext/ciphertext pairs (**known plaintext attack**)
  c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker (**chosen plaintext attack**)
  d) collection of plaintext/ciphertext pairs for ciphertexts selected by the attacker (**chosen ciphertext attack**)



Brute-Force Attack

- Try all possible keys K and determine if $D_K(C)$ is a likely plaintext
  - Requires some knowledge of the structure of the plaintext (e.g., PDF file or email message)
- Key should be a sufficiently long random value to make exhaustive search attacks unfeasible

Image by Michael Cote from http://commons.wikimedia.org/wiki/File:Bingo_cards.jpg

Classical Cryptography

- Transposition Cipher
- Substitution Cipher
  - Simple substitution cipher (Caesar cipher)
  - Vigenere cipher
  - One-time pad

https://emn178.github.io/online-tools/sha1.html

Transposition Cipher: rail fence

- Write plaintext in two rows
- Generate ciphertext in column order

- Example: “HELLOWORLD”

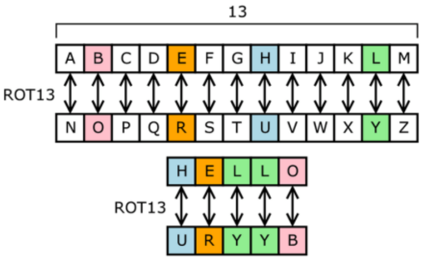
HLOOL
ELWRD
ciphertext: HLOOLELWRD

Problem: does not affect the frequency of individual symbols

Substitution Ciphers

- Each letter is uniquely replaced by another.
- There are 26! possible substitution ciphers for English language.
- There are more than 4.03 x 10^{26} such ciphers.

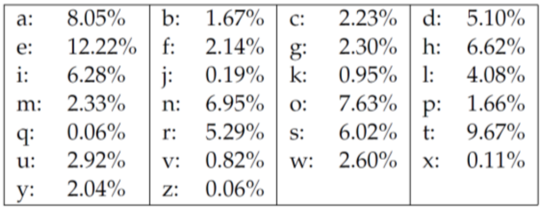
- One popular substitution “cipher” for some Internet posts is ROT13.



Public domain image from http://en.wikipedia.org/wiki/File:ROT13.png

Frequency Analysis

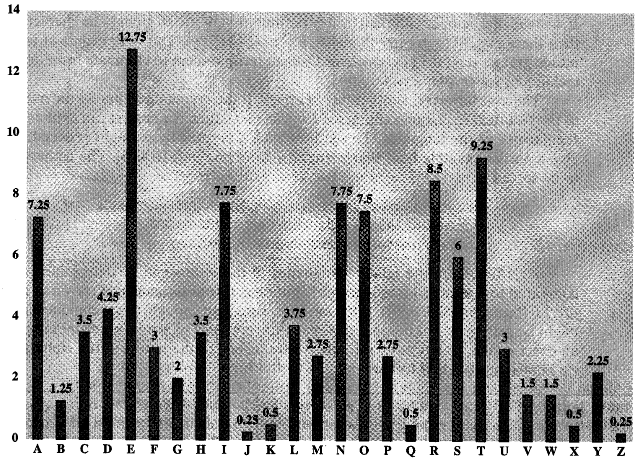
- Letters in a natural language, like English, are not uniformly distributed.
- Knowledge of letter frequencies, including pairs and triples can be used in cryptologic attacks against substitution ciphers.



a:	8.05%	b:	1.67%	c:	2.23%	d:	5.10%
e:	12.22%	f:	2.14%	g:	2.30%	h:	6.62%
i:	6.28%	j:	0.19%	k:	0.95%	l:	4.08%
m:	2.33%	n:	6.95%	o:	7.63%	p:	1.66%
q:	0.06%	r:	5.29%	s:	6.02%	t:	9.67%
u:	2.92%	v:	0.82%	w:	2.60%	x:	0.11%
y:	2.04%	z:	0.06%				

Letter frequencies in the book *The Adventures of Tom Sawyer*, by Twain.

Distribution of Letters in English

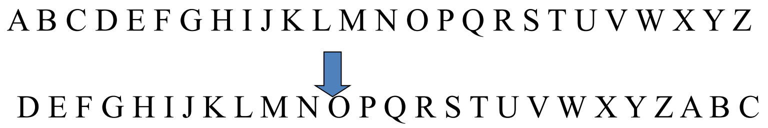


Frequency analysis

Simple substitution cipher

substituting one thing for another

- Simplest one: monoalphabetic cipher:
  - substitute one letter for another (**Caesar Cipher**)



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

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

Example: encrypt "I attack"

Vigenere Cipher

- Idea: Uses Caesar's cipher with various different shifts, in order to hide the distribution of the letters.
- A key defines the shift used in each letter in the text
- A key word is repeated as many times as required to become the same length

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Plain text: I a t t a c k
Key: 2 3 4 2 3 4 2
Cipher text: K d x v d g m

(key is "234")

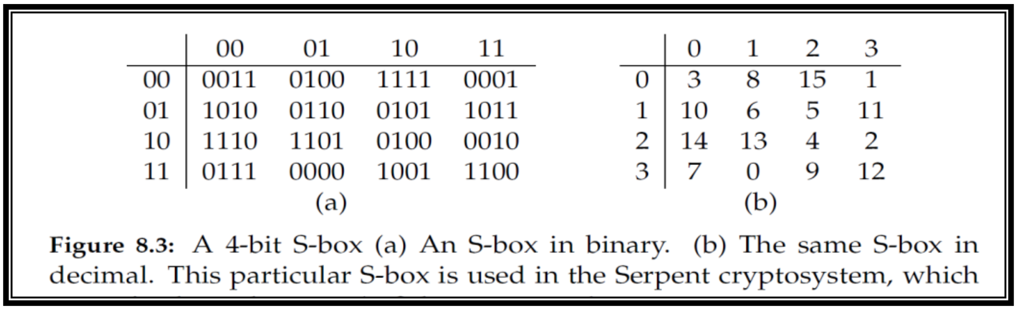
Problem of Vigenere Cipher

- Vigenere is easy to break (Kasiski, 1863):
- Assume we know the length of the key. We can organize the ciphertext in rows with the same length of the key. Then, every column can be seen as encrypted using Caesar's cipher.

- The length of the key can be found using several methods:
  - 1. If short, try 1, 2, 3, ...
  - 2. Find repeated strings in the ciphertext. Their distance is expected to be a multiple of the length. Compute the GCD of (most) distances.
  - 3. Use the index of coincidence.

Substitution Boxes

- Substitution can also be done on binary numbers.
- Such substitutions are usually described by substitution boxes, or S-boxes.



	00	01	10	11
00	0011	0100	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

(a)

	0	1	2	3
0	3	8	15	1
1	10	6	5	11
2	14	13	4	2
3	7	0	9	12

(b)

Figure 8.3: A 4-bit S-box (a) An S-box in binary. (b) The same S-box in decimal. This particular S-box is used in the Serpent cryptosystem, which

Example:

One good example of a fixed table is the S-box from DES (S_5), mapping 6-bit input into a 4-bit output: Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits (the first and last bits), and the column using the inner four bits. For example, an input "**011011**" has outer bits "**01**" and inner bits "1101"; the corresponding output would be "1001“

Total Average from (000000)-(111111)

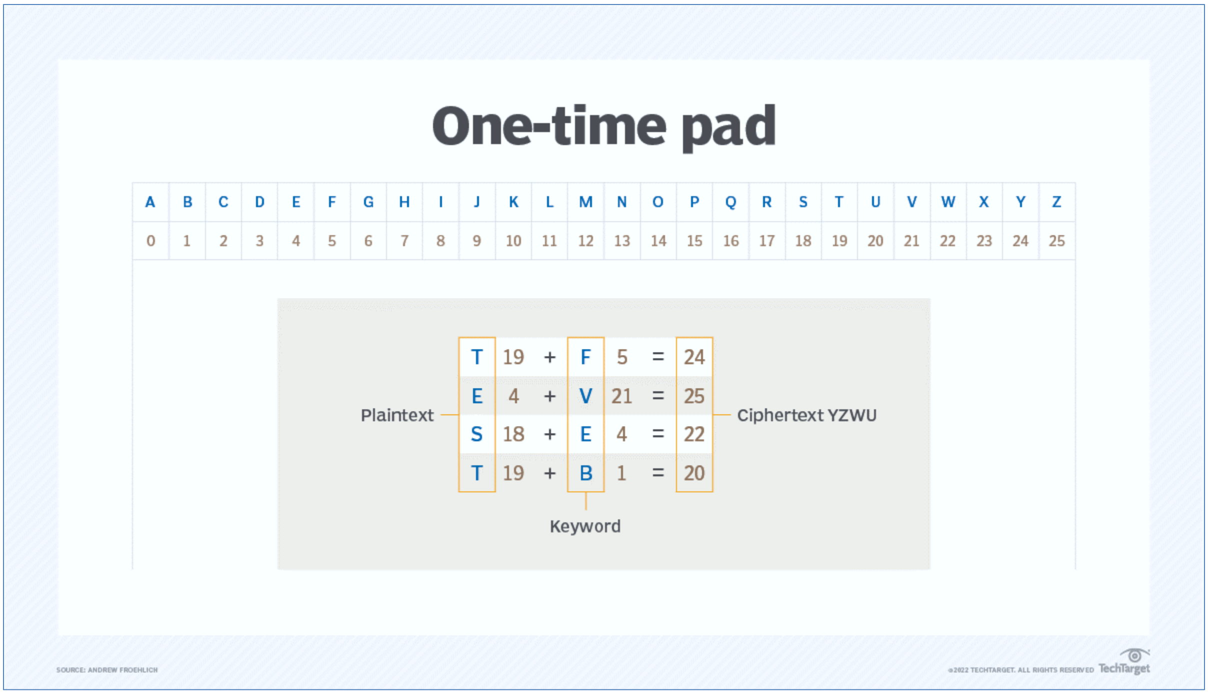
S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

https://www.tutorialspoint.com/what-is-s-box-substitution

One-Time Pads

- Extended from Vigenère cipher
- There is one type of substitution cipher that is absolutely unbreakable.
  - The **one-time pad** was invented in 1917 by Joseph Mauborgne and Gilbert Vernam
  - We use a block of shift keys, $(k_1, k_2, \ldots, k_n)$, to encrypt a plaintext, M, of length n, with each shift key being chosen uniformly at random.
- Since each shift is random, every ciphertext is equally likely for any plaintext.

- **One-time pad cipher is a type of Vignere cipher which includes the following features:**
  - It is an unbreakable cipher.
  - The key is exactly same as the length of message which is encrypted.
  - The key is made up of random symbols.
  - As the name suggests, key is used one time only and never used again for any other message to be encrypted.
- **Why is it Unbreakable?**
  - The key is unbreakable owing to the following features −
  - The key is as long as the given message.
  - The key is truly random and specially auto-generated.
  - Each key should be used once and destroyed by both sender and receiver.
  - There should be two copies of key: one with the sender and other with the receiver.
- `



One-time pad

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

T	19	+	F	5	=	24
E	4	+	V	21	=	25
S	18	+	E	4	=	22
T	19	+	B	1	=	20

SOURCE: ANDREW FROEHLICH

©2022 TECHTARGET. ALL RIGHTS RESERVED TechTarget

Weaknesses of the One-Time Pad

- In spite of their perfect security, one-time pads have some weaknesses
- The key has to be as long as the plaintext
- Keys can never be reused
  - Repeated use of one-time pads allowed the U.S. to break some of the communications of Soviet spies during the Cold War.

Plaintext

MD5

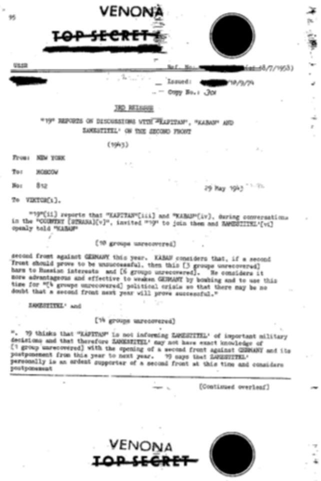
Hashed Code

Key

MD5

Hashed Key

Onetime Pad



VENONA

~~TOP SECRET~~ [redacted]

USSR [redacted]

Ref. No.: [redacted] (of 18/7/1953)

Issued: [redacted] 10/9/74

Copy No.: 304

3RD RELEASE

"19" REPORTS ON DISCUSSIONS WITH "KAPITAN", "KABAN" AND ZAMESTITEL' ON THE SECOND FRONT

(1943)

From: NEW YORK

To: MOSCOW

No: 812

29 May 1943

To VIKTOR[i].

"19"[ii] reports that "KAPITAN"[iii] and "KABAN"[iv], during conversations in the "COUNTRY [STRANA][v]", invited "19" to join them and ZAMESTITEL'[vi] openly told "KABAN"

[10 groups unrecovered]

second front against GERMANY this year. KABAN considers that, if a second front should prove to be unsuccessful, then this [3 groups unrecovered] harm to Russian interests and [6 groups unrecovered]. He considers it more advantageous and effective to weaken GERMANY by bombing and to use this time for "[4 groups unrecovered] political crisis so that there may be no doubt that a second front next year will prove successful."

ZAMESTITEL' and

[14 groups unrecovered]

". 19 thinks that "KAPITAN" is not informing ZAMESTITEL' of important military decisions and that therefore ZAMESTITEL' may not have exact knowledge of [1 group unrecovered] with the opening of a second front against GERMANY and its postponement from this year to next year. 19 says that ZAMESTITEL' personally is an ardent supporter of a second front at this time and considers postponement

[Continued overleaf]

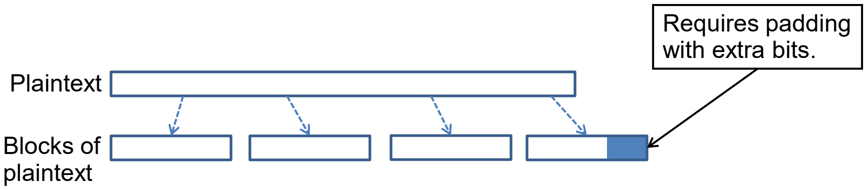
VENONA

~~TOP SECRET~~

Public domain declassified government image from https://www.cia.gov/library/center-for-the-study-of-intelligence/csi-publications/books-and-monographs/venona-soviet-espionage-and-the-american-response-1939-1957/part2.htm

Block Ciphers

- In a **Block cipher:**
  - Plaintext and ciphertext have **fixed length b** (e.g., 128 bits)
  - A plaintext of length n is partitioned into a sequence of m **blocks**, P[0], …, P[m–1], **where n ≤ bm < n + b**
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks.



Requires padding with extra bits.

Plaintext

Blocks of plaintext

Padding

- Block ciphers require the length n of the plaintext to be a multiple of the block size b
- Padding the last block needs to be unambiguous (cannot just add zeroes)
- When the block size and plaintext length are a multiple of 8, a common padding method (PKCS5) is a sequence of identical bytes, each indicating the length (in bytes) of the padding
- Example for b = 128 (16 bytes)
  - Plaintext: “Roberto” (7 bytes)
  - Padded plaintext: “Roberto999999999” (16 bytes), where 9 denotes the number and not the character
- **We need to always pad the last block, which may consist only of padding**

Block Ciphers in Practice

- **Data Encryption Standard (DES)**
  - Developed by IBM and adopted by NIST in 1977
  - ***64-bit blocks and 56-bit keys***
  - Small key space makes exhaustive search attack feasible since late 90s
- **Triple DES (3DES)**
  - Nested application of DES with three different keys KA, KB, and KC
  - Effective ***key length is 168 bits***, making exhaustive search attacks unfeasible
  - $C = E_{KC}(D_{KB}(E_{KA}(P)))$; $P = D_{KA}(E_{KB}(D_{KC}(C)))$
  - Equivalent to DES when KA=KB=KC (backward compatible)
- **Advanced Encryption Standard (AES)**
  - Selected by NIST in 2001 through open international competition and public discussion
  - ***128-bit blocks and several possible key lengths: 128, 192 and 256 bits***
  - Exhaustive search attack not currently possible
  - AES-256 is the symmetric encryption algorithm of choice

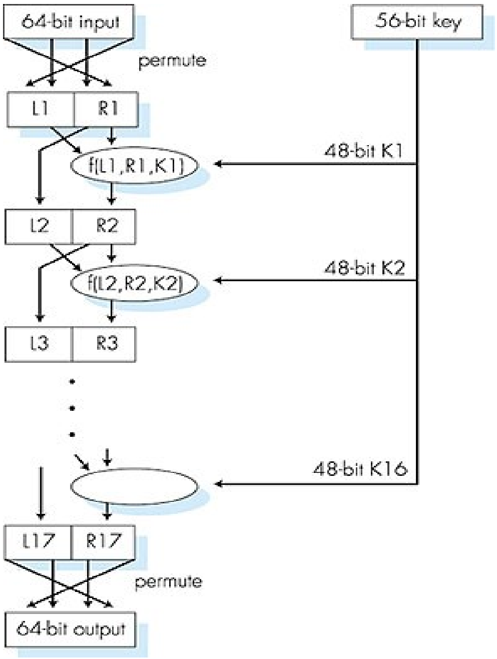
Symmetric key crypto: DES

DES operation

initial permutation

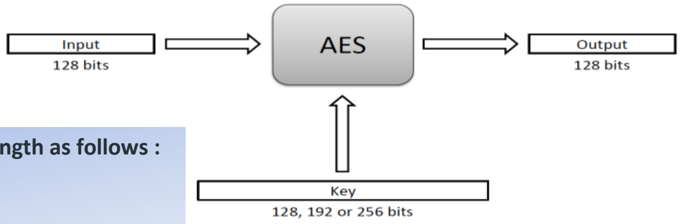
16 identical “rounds” of function application, each using different 48 bits of key

final permutation



The Advanced Encryption Standard (AES)

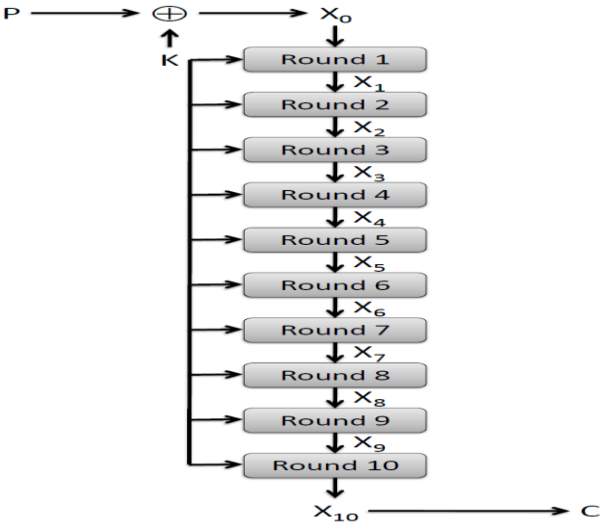
- In 1997, the U.S. National Institute for Standards and Technology (NIST) put out a public call for a replacement to DES.
- It narrowed down the list of submissions to five finalists, and ultimately chose an algorithm that is now known as the **Advanced Encryption Standard** (**AES**).
- AES is a block cipher that operates on 128-bit blocks as I/O. It is designed to be used with keys that are 128, 192, or 256 bits long, yielding ciphers known as AES-128, AES-192, and AES-256.



The number of rounds depends on the key length as follows :
•128 bit key – 10 rounds
•192 bit key – 12 rounds
•256 bit key – 14 rounds

AES Round Structure

- The 128-bit version of the AES encryption algorithm proceeds in ten rounds.
- Each round performs an invertible transformation on a 128-bit array, called **state**.
- The initial state X_0 is the XOR of the plaintext P with the key K:
- X_0 = P XOR K.
- Round i (i = 1, …, 10) receives state X_{i-1} as input and produces state X_i.
- The ciphertext C is the output of the final round: C = X_{10}.



AES Rounds

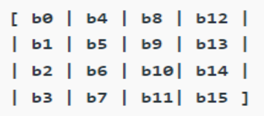
Each round is built from four basic steps:

1. **SubBytes step**: an S-box substitution step
2. **ShiftRows step**: a permutation step
3. **MixColumns step**: a matrix multiplication step
4. **AddRoundKey step**: an XOR step with a **round key** derived from the 128-bit encryption key

The last round doesn’t have the MixColumns round.

Encryption:

AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.



```
[ b0 | b4 | b8 | b12 |
| b1 | b5 | b9 | b13 |
| b2 | b6 | b10| b14 |
| b3 | b7 | b11| b15 ]
```

SubBytes:

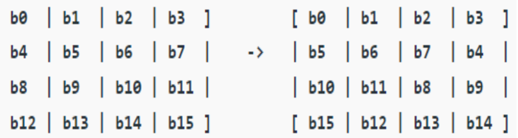
This step implements the substitution. In this step each byte is substituted by another byte. Its performed using a **lookup table also called the S-box**. This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16 byte (4 x 4) matrix like before.

ShiftRows :

This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

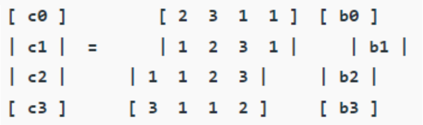
(A left circular shift is performed.)



```
[ b0  | b1  | b2  | b3  ]        [ b0  | b1  | b2  | b3  ]
| b4  | b5  | b6  | b7  |   ->   | b5  | b6  | b7  | b4  |
| b8  | b9  | b10 | b11 |        | b10 | b11 | b8  | b9  |
[ b12 | b13 | b14 | b15 ]        [ b15 | b12 | b13 | b14 ]
```

MixColumns:

This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.



```
[ c0 ]        [ 2  3  1  1 ]  [ b0 ]
| c1 |  =     | 1  2  3  1 |     | b1 |
| c2 |     | 1  1  2  3 |     | b2 |
[ c3 ]     [ 3  1  1  2 ]     [ b3 ]
```

Decryption :

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes.Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

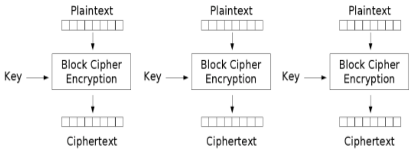
The stages of each round in decryption is as follows :

- Add round key
- Inverse MixColumns
- ShiftRows
- Inverse SubByte

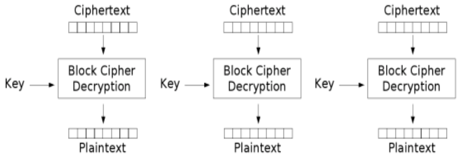
https://www.educative.io/answers/what-is-the-aes-algorithm

Block Cipher Modes

- A block cipher mode describes the way a block cipher encrypts and decrypts a sequence of message blocks.
- Electronic Code Book (ECB) Mode (is the simplest):
  - Block P[i] encrypted into ciphertext block C[i] = $E_K(P[i])$
  - Block C[i] decrypted into plaintext block M[i] = $D_K(C[i])$



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

Public domain images from http://en.wikipedia.org/wiki/File:Ecb_encryption.png and http://en.wikipedia.org/wiki/File:Ecb_decryption.png

Strengths and Weaknesses of ECB

- **Strengths**:
  - Is very simple
  - Allows for parallel encryptions of the blocks of a plaintext
  - Can tolerate the loss or damage of a block

- **Weakness**:
  - Documents and images are not suitable for ECB encryption since patters in the plaintext are repeated in the ciphertext:

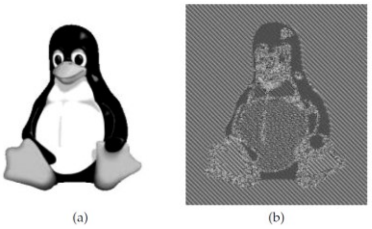
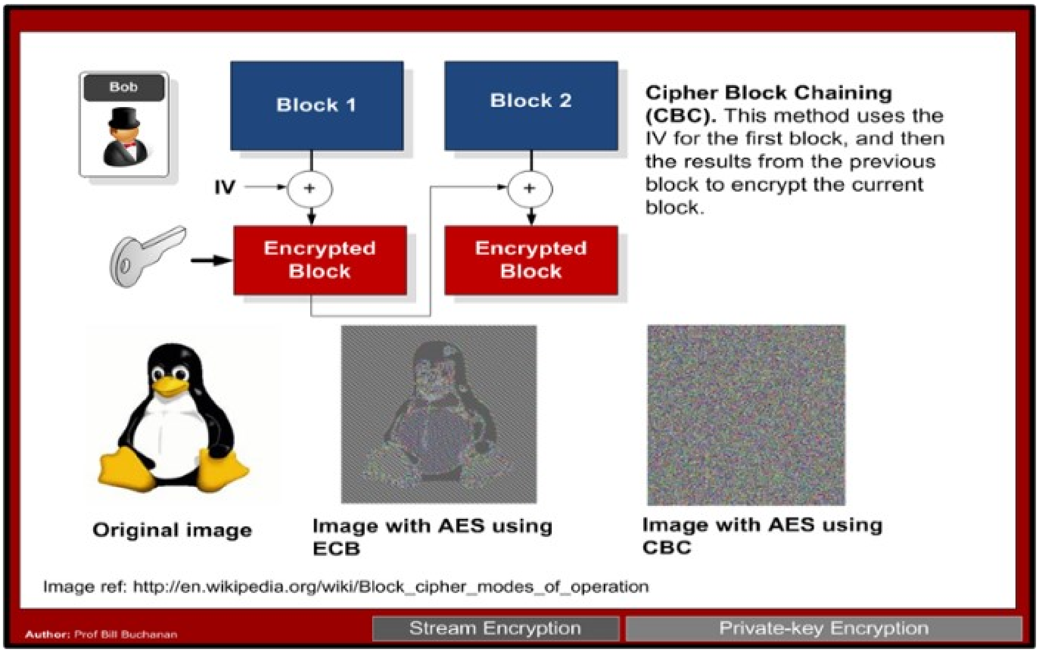


Figure 8.6: How ECB mode can leave identifiable patterns in a sequence of blocks: (a) An image of Tux the penguin, the Linux mascot. (b) An encryption of the Tux image using ECB mode. (The image in (a) is by Larry Ewing, lewing@isc.tamu.edu, using The Gimp; the image in (b) is by Dr. Juzam. Both are used with permission via attribution.)



Cipher Block Chaining (CBC). This method uses the IV for the first block, and then the results from the previous block to encrypt the current block.

Original image

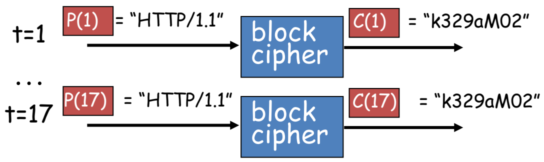
Image with AES using ECB

Image with AES using CBC

Image ref: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

Author: Prof Bill Buchanan

Another Example



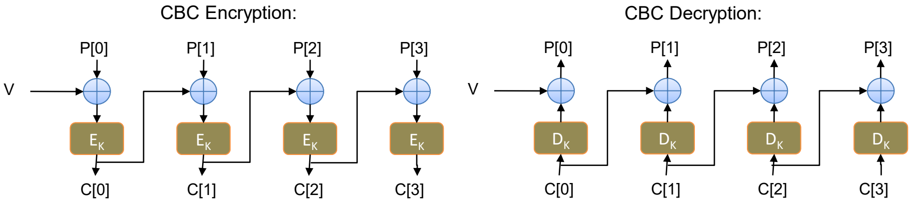
https://www.base64-image.de/

https://base64.guru/converter/decode/image

https://www.devglan.com/online-tools/aes-encryption-decryption

Cipher Block Chaining (CBC) Mode

- In Cipher Block Chaining (CBC) Mode
  - The previous ciphertext block is combined with the current plaintext block $C[i] = E_K(C[i-1] \oplus P[i])$
  - $C[-1] = V$, a random block separately transmitted encrypted (known as the initialization vector)
  - Decryption: $P[i] = C[i-1] \oplus D_K(C[i])$



Strengths and Weaknesses of CBC

- Strengths:
  - Doesn’t show patterns in the plaintext
  - Is the most common mode
  - Is fast and relatively simple

- Weaknesses:
  - CBC requires the reliable transmission of all the blocks sequentially
  - CBC is not suitable for applications that allow packet losses (e.g., music and video streaming)

Hill Cipher: a cipher based on matrix multiplication

- Message P =“ACTDOG”, use m=3
  - Break into two blocks: “ACT”, and “DOG”
  - 'A' is 0, 'C' is 2 and 'T' is 19, “ACT” is the vector: x= $\begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}$
  - Encryption key is a 3*3 matrix: K= $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$
  - The cipher text of the first block is:

c = K·x $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} = \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} \equiv \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26}$ = $\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$

c =‘POH’

Hill Cipher

- If the first block plaintext is ‘CAT’
  - x = $\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$
  - c=K · x $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} \equiv \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} \equiv \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} \pmod{26}$
  - c= ‘FIN’
  - The Hill cipher has achieved Shannon's diffusion, and an n-dimensional Hill cipher can diffuse fully across n symbols at once.
  - This and the previous slide’s examples are from Wikipedia http://en.wikipedia.org/wiki/Hill_cipher

Hill Cipher Decryption

- x= $K^{-1} \cdot c$

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}^{-1} \equiv \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \pmod{26}$$

- Its features:
  - Interpret letters as numbers
  - Linear algebra
- Both features have been used in DES and AES

Hill Cipher to Realize Transposition

- Transposition can be realized by special Hill cipher
- Special key K = $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Then it corresponds to permutation:
  - π: (1,2,3) → (2,1,3)

Stream Cipher

- **Key stream**
  - Pseudo-random sequence of bits S = S[0], S[1], S[2], …
  - Can be generated on-line one bit (or byte) at the time
- **Stream cipher**
  - XOR the plaintext with the key stream C[i] = S[i] ⊕ P[i]
  - Suitable for plaintext of arbitrary length generated on the fly, e.g., media stream
- **Synchronous stream cipher**
  - Key stream obtained only from the secret key K
    - Independent with plaintext and ciphertext
  - Works for high-error channels if plaintext has packets with sequence numbers
  - Sender and receiver must synchronize in using key stream
  - If a digit is corrupted in transmission, only a single digit in the plaintext is affected and the error does not propagate to other parts of the message.

- Self-synchronizing stream cipher
  - Key stream obtained from the secret key and N previous ciphertexts
  - the receiver will automatically synchronize with the keystream generator after receiving *N* ciphertext digits, making it easier to recover if digits are dropped or added to the message stream.
  - Lost packets cause a delay of q steps before decryption resumes
  - Single-digit errors are limited in their effect, affecting only up to *N* plaintext digits.

Key Stream Generation

- **RC4**
  - Designed in 1987 by Ron Rivest for RSA Security
  - Trade secret until 1994
  - Uses keys with up to 2,048 bits
  - Simple algorithm
- **Block cipher in counter mode (CTR)**
  - Use a block cipher with block size b
  - The secret key is a pair (K,t), where K is key and t (counter) is a b-bit value
  - The key stream is the concatenation of ciphertexts

    $$E_K(t), E_K(t + 1), E_K(t + 2), \ldots$$
  - Can use a shorter counter concatenated with a random value
  - Synchronous stream cipher

Attacks on Stream Ciphers

- Repetition attack
  - if key stream reused, attacker obtains XOR of two plaintexts (why?)

Public Key Encryption

Public Key Cryptography

symmetric key crypto

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?
  - Typical chicken and egg dilemma.

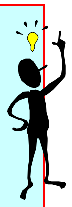
public key cryptography

r radically different approach [Diffie-Hellman76, RSA78]

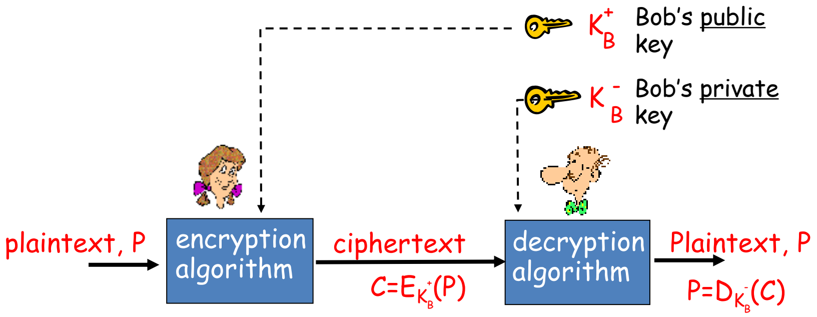
r sender, receiver do *not* share secret key

r *public* encryption key known *to all*

r *private* decryption key known only to receiver



Public key cryptography



K_B^+ Bob's public key

K_B^- Bob's private key

plaintext, P

encryption algorithm

ciphertext

$C=E_{K_B^+}(P)$

decryption algorithm

Plaintext, P

$P=D_{K_B^-}(C)$

Facts About Numbers

- Prime number $\boldsymbol{p}$:
  - $\boldsymbol{p}$ is an integer
  - $\boldsymbol{p} \geq 2$
  - The only divisors of $\boldsymbol{p}$ are 1 and $\boldsymbol{p}$
- Examples
  - $2, 7, 19$ are primes
  - $-3, 0, 1, 6$ are not primes
- Prime decomposition of a positive integer $\boldsymbol{n}$:

$$\boldsymbol{n} = \boldsymbol{p}_1^{\boldsymbol{e}_1} \times \ldots \times \boldsymbol{p}_{\boldsymbol{k}}^{\boldsymbol{e}_{\boldsymbol{k}}}$$

- Example:
  - $200 = 2^3 \times 5^2$

Fundamental Theorem of Arithmetic

The prime decomposition of a positive integer is unique

Greatest Common Divisor

- The greatest common divisor (GCD) of two positive integers $\boldsymbol{a}$ and $\boldsymbol{b}$, denoted $\gcd(\boldsymbol{a}, \boldsymbol{b})$, is the largest positive integer that divides both $\boldsymbol{a}$ and $\boldsymbol{b}$
- The above definition is extended to arbitrary integers
- Examples:

$\gcd(18, 30) = 6$ $\gcd(0, 20) = 20$

$\gcd(-21, 49) = 7$

- Two integers a and b are said to be relatively prime if

$$\gcd(\boldsymbol{a}, \boldsymbol{b}) = 1$$

- Example:
  - Integers 15 and 28 are relatively prime

Modular Arithmetic

- Modulo operator for a positive integer $\boldsymbol{n}$

  $$\boldsymbol{r} = \boldsymbol{a} \bmod \boldsymbol{n}$$

  equivalent to

  $$\boldsymbol{a} = \boldsymbol{r} + \boldsymbol{kn}$$

  and

  $$\boldsymbol{r} = \boldsymbol{a} - \lfloor \boldsymbol{a}/\boldsymbol{n} \rfloor \boldsymbol{n}$$

- Example:

  $29 \bmod 13 = 3$ $\quad$ $13 \bmod 13 = 0$ $\quad$ $-1 \bmod 13 = 12$

  $29 = 3 + 2\times13$ $\quad$ $13 = 0 + 1\times13$ $\quad$ $12 = -1 + 1\times13$

  For a<0, we first add a large kn to a such that it becomes positive

- Modulo and GCD:

  $$\gcd(\boldsymbol{a}, \boldsymbol{b}) = \gcd(\boldsymbol{b}, \boldsymbol{a} \bmod \boldsymbol{b})$$

- Example:

  $\gcd(21, 12) = 3$ $\quad$ $\gcd(12, 21 \bmod 12) = \gcd(12, 9) = 3$

Euclid’s GCD Algorithm

- Euclid’s algorithm for computing the GCD repeatedly applies the formula

  $$\gcd(\boldsymbol{a}, \boldsymbol{b}) = \gcd(\boldsymbol{b}, \boldsymbol{a} \bmod \boldsymbol{b})$$

- Example
  - $\gcd(412, 260) = 4$

```
Algorithm EuclidGCD(a, b)
  Input integers a and b
  Output gcd(a, b)

  if $b = 0$
    return a
  else
    return EuclidGCD(b, a mod b)
```

a	412	260	152	108	44	20	4
b	260	152	108	44	20	4	0

RSA: Choosing keys

1. Choose two large prime numbers *p, q.* (e.g., 1024 bits each)
2. Compute $n = pq$, $z = (p-1)(q-1)$
3. Choose e (with $e<n$) that has no common factors with z. (*e, z* are "relatively prime").
4. Choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5. *Public* key is $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: Encryption, decryption

0. Given (*n,e*) and (*n,d*) as computed above
1. To encrypt bit pattern, *m*, compute
   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by *n*)
2. To decrypt received bit pattern, *c*, compute
   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by *n*)

Magic happens! $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

RSA example:

Bob chooses $p=5, q=7$. Then $n=35, z=24$.
$e=5$ (so e, z relatively prime).
$d=29$ (so $ed-1$ exactly divisible by z).

	letter	m	m^e	$c = m^e \bmod n$
encrypt:	l	12	1524832	17

	c	c^d	$m = c^d \bmod n$	letter
decrypt:	17	481968572106750915091411825223071697	12	l

Computational extensive

RSA: Why is that $m = (m^e \bmod n)^d \bmod n$

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

RSA: another important property

The following property will be *very* useful later:

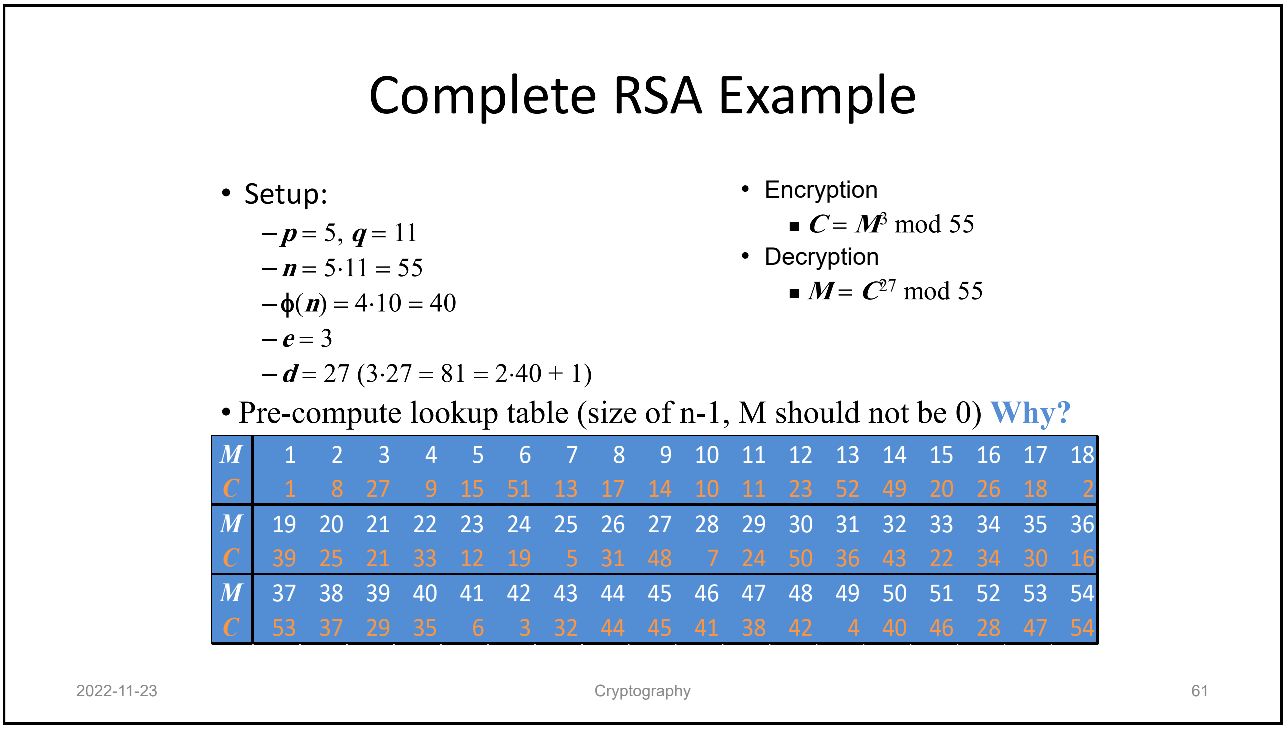
$$\underbrace{E_{K^+}(E_{K^-}(P))}_{\substack{\text{use public key}\\ \text{first, followed}\\ \text{by private key}}} = P = \underbrace{E_{K^-}(E_{K^+}(P))}_{\substack{\text{use private key}\\ \text{first, followed}\\ \text{by public key}}}$$

Result is the same!

RSA Cryptosystem

- Setup:
  - $n = pq$, with p and q primes
  - e relatively prime to $\phi(n) = (p - 1)(q - 1)$
  - d inverse of e in $Z_{\phi(n)}$
    - *ed* mod *z* = *1*
- Keys:
  - Public key: $K_E = (n, e)$
  - Private key: $K_D = d$
- Encryption:
  - Plaintext M in Z_n
  - $C = M^e \bmod n$
- Decryption:
  - $M = C^d \bmod n$

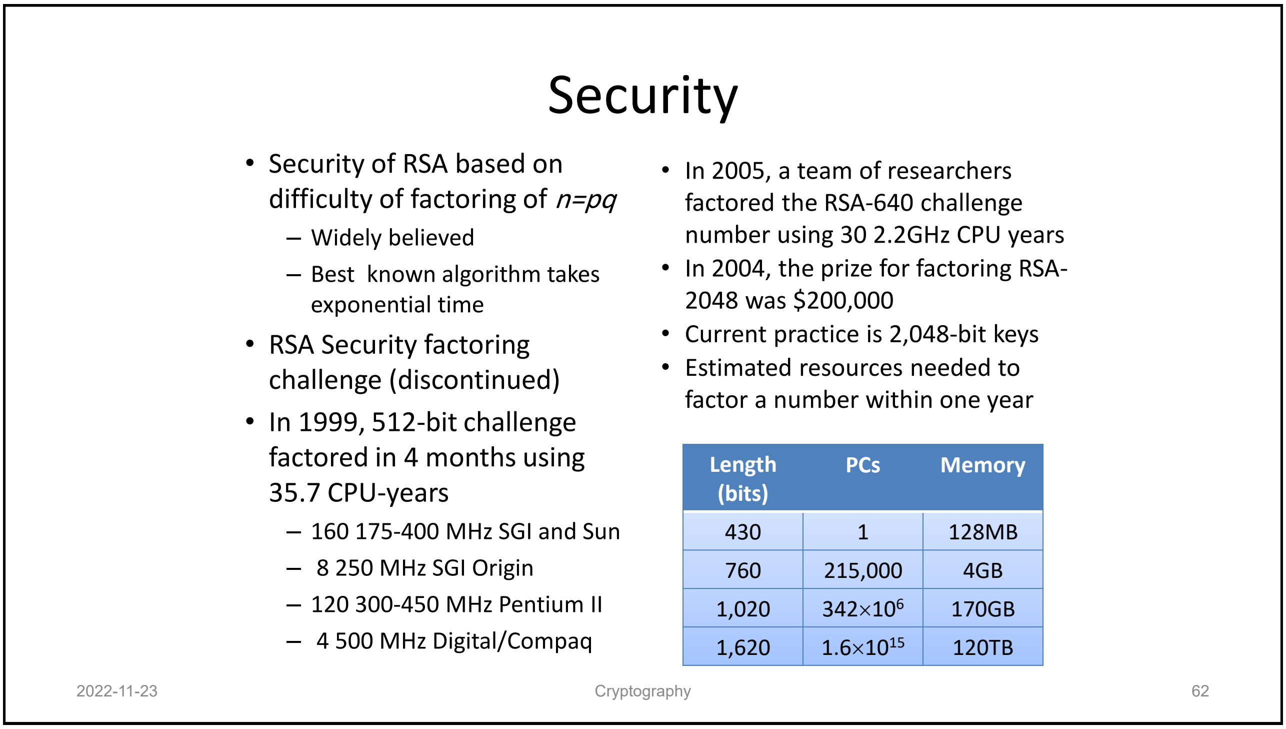
- Example
  - Setup:
    - $p = 7,\ q = 17$
    - $n = 7 \cdot 17 = 119$
    - $\phi(n) = 6 \cdot 16 = 96$
    - $e = 5$
    - $d = 77$
  - Keys:
    - public key: (119, 5)
    - private key: 77
  - Encryption:
    - $M = 19$
    - $C = 19^5 \bmod 119 = 66$
  - Decryption:
    - $C = 66^{77} \bmod 119 = 19$



Complete RSA Example

- Setup:
  - $p = 5$, $q = 11$
  - $n = 5\cdot11 = 55$
  - $\phi(n) = 4\cdot10 = 40$
  - $e = 3$
  - $d = 27$ $(3\cdot27 = 81 = 2\cdot40 + 1)$
- Encryption
  - $C = M^3 \bmod 55$
- Decryption
  - $M = C^{27} \bmod 55$
- Pre-compute lookup table (size of n-1, M should not be 0) **Why?**

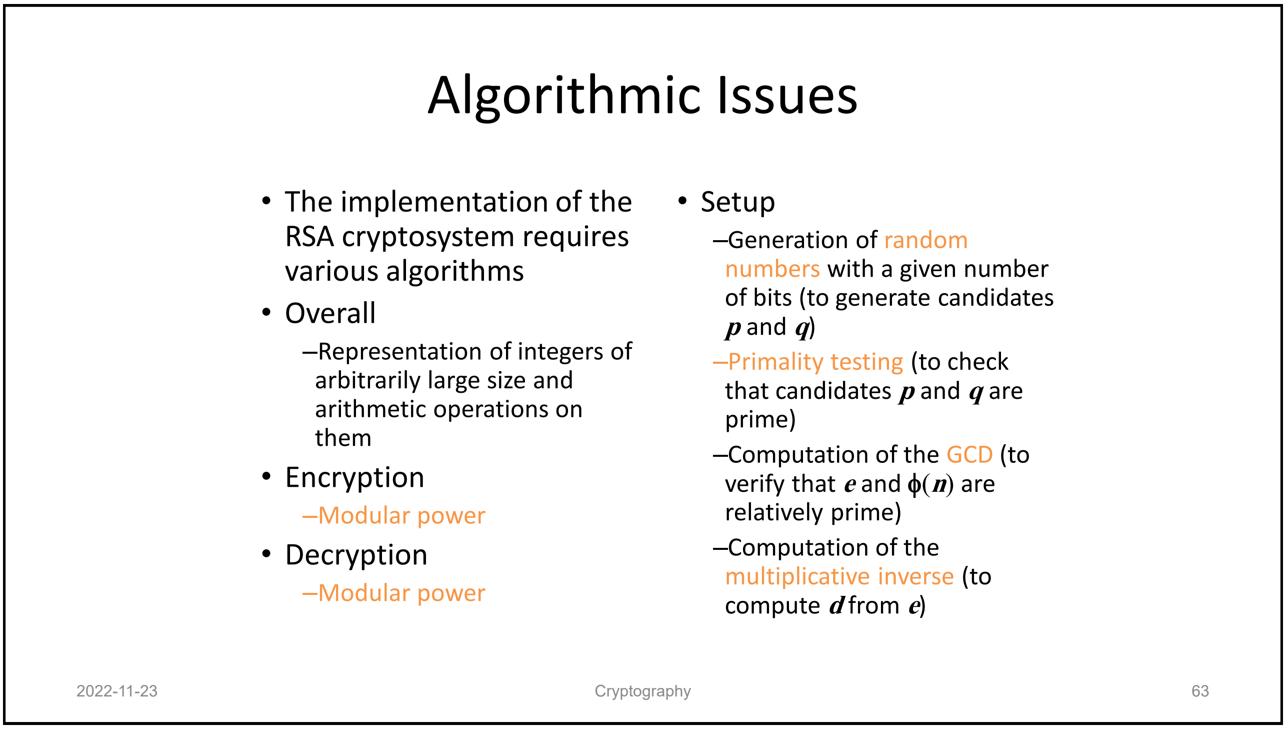
M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
C	1	8	27	9	15	51	13	17	14	10	11	23	52	49	20	26	18	2
M	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
C	39	25	21	33	12	19	5	31	48	7	24	50	36	43	22	34	30	16
M	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
C	53	37	29	35	6	3	32	44	45	41	38	42	4	40	46	28	47	54



Security

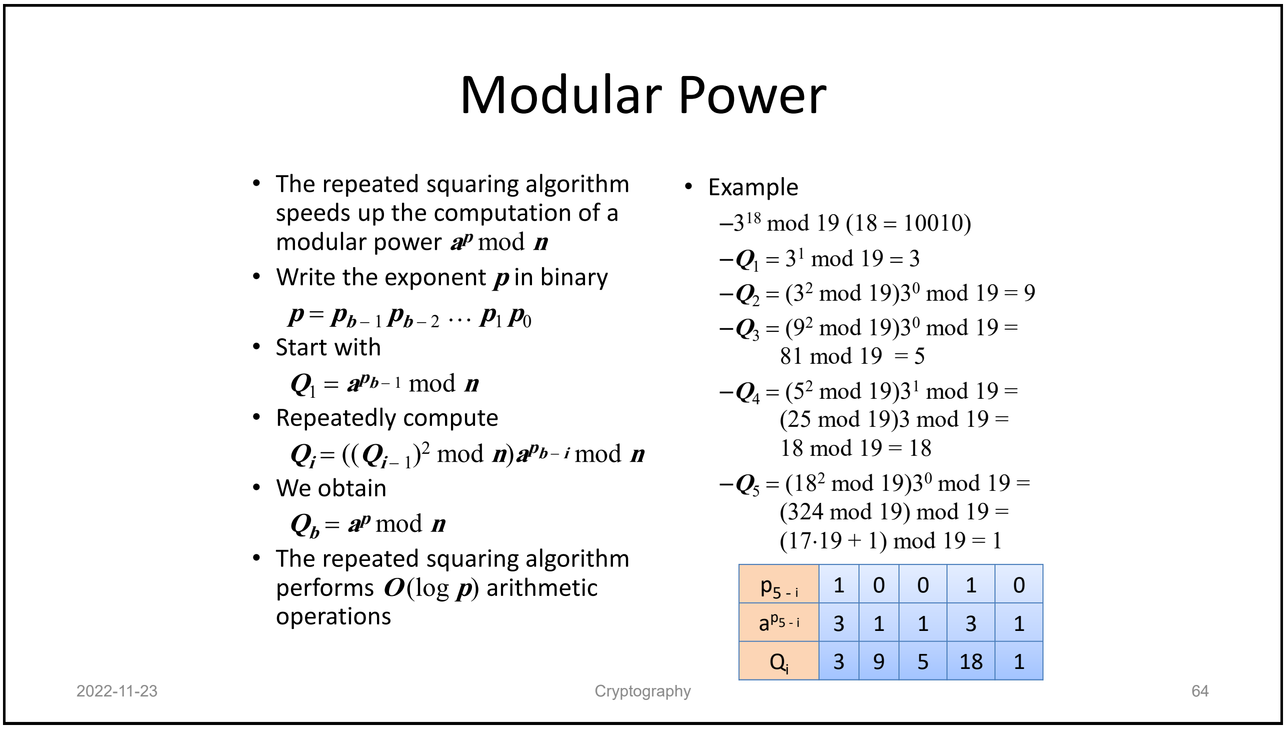
- Security of RSA based on difficulty of factoring of *n=pq*
  - Widely believed
  - Best known algorithm takes exponential time
- RSA Security factoring challenge (discontinued)
- In 1999, 512-bit challenge factored in 4 months using 35.7 CPU-years
  - 160 175-400 MHz SGI and Sun
  - 8 250 MHz SGI Origin
  - 120 300-450 MHz Pentium II
  - 4 500 MHz Digital/Compaq
- In 2005, a team of researchers factored the RSA-640 challenge number using 30 2.2GHz CPU years
- In 2004, the prize for factoring RSA-2048 was $200,000
- Current practice is 2,048-bit keys
- Estimated resources needed to factor a number within one year

Length (bits)	PCs	Memory
430	1	128MB
760	215,000	4GB
1,020	342×10^6	170GB
1,620	1.6×10^{15}	120TB



Algorithmic Issues

- The implementation of the RSA cryptosystem requires various algorithms
- Overall
  - Representation of integers of arbitrarily large size and arithmetic operations on them
- Encryption
  - Modular power
- Decryption
  - Modular power
- Setup
  - Generation of random numbers with a given number of bits (to generate candidates $\boldsymbol{p}$ and $\boldsymbol{q}$)
  - Primality testing (to check that candidates $\boldsymbol{p}$ and $\boldsymbol{q}$ are prime)
  - Computation of the GCD (to verify that $\boldsymbol{e}$ and $\phi(\boldsymbol{n})$ are relatively prime)
  - Computation of the multiplicative inverse (to compute $\boldsymbol{d}$ from $\boldsymbol{e}$)



Modular Power

- The repeated squaring algorithm speeds up the computation of a modular power $\boldsymbol{a}^{\boldsymbol{p}} \bmod \boldsymbol{n}$
- Write the exponent $\boldsymbol{p}$ in binary

  $\boldsymbol{p} = \boldsymbol{p}_{\boldsymbol{b}-1}\boldsymbol{p}_{\boldsymbol{b}-2} \cdots \boldsymbol{p}_1 \boldsymbol{p}_0$
- Start with

  $\boldsymbol{Q}_1 = \boldsymbol{a}^{\boldsymbol{p}_{\boldsymbol{b}-1}} \bmod \boldsymbol{n}$
- Repeatedly compute

  $\boldsymbol{Q}_{\boldsymbol{i}} = ((\boldsymbol{Q}_{\boldsymbol{i}-1})^2 \bmod \boldsymbol{n})\boldsymbol{a}^{\boldsymbol{p}_{\boldsymbol{b}-\boldsymbol{i}}} \bmod \boldsymbol{n}$
- We obtain

  $\boldsymbol{Q}_{\boldsymbol{b}} = \boldsymbol{a}^{\boldsymbol{p}} \bmod \boldsymbol{n}$
- The repeated squaring algorithm performs $\boldsymbol{O}(\log \boldsymbol{p})$ arithmetic operations
- Example
  - $3^{18} \bmod 19$ $(18 = 10010)$
  - $\boldsymbol{Q}_1 = 3^1 \bmod 19 = 3$
  - $\boldsymbol{Q}_2 = (3^2 \bmod 19)3^0 \bmod 19 = 9$
  - $\boldsymbol{Q}_3 = (9^2 \bmod 19)3^0 \bmod 19 =$ $81 \bmod 19 = 5$
  - $\boldsymbol{Q}_4 = (5^2 \bmod 19)3^1 \bmod 19 =$ $(25 \bmod 19)3 \bmod 19 =$ $18 \bmod 19 = 18$
  - $\boldsymbol{Q}_5 = (18^2 \bmod 19)3^0 \bmod 19 =$ $(324 \bmod 19) \bmod 19 =$ $(17 \cdot 19 + 1) \bmod 19 = 1$

p_{5-i}	1	0	0	1	0
$a^{p_{5-i}}$	3	1	1	3	1
Q_i	3	9	5	18	1

Example of RSA Algorithm

- Choose two large prime numbers P and Q, Let P = 7, Q = 17
- Calculate N = P x Q , We have, N = 7 x 17 = 119.
- Choose the public key (i.e., the encryption key) **E** such that it is not an element of (P -1) x (Q – 1)
  - Let us find (7 - 1) x (17 -1) = 6 x 16 = 96
  - The factors of 96 are 2, 2, 2, 2, 2, and 3 (because 96 = 2 x 2 x 2 x 2 x 2 x 3).
  - Therefore, it can select E such that none of the factors of E is 2 and 3.
  - We cannot choose E as 4 (because it has 2 as a factor), 15 (because it has 3 as a factor) and 6 (because it has 2 and 3 both as factors).
  - **Let us choose E as 5 (it can have been any other number that does not its factors as 2 and 3).**
- Choose the private key (i.e., the decryption key) D including the following equation is true: (D x E) mod (P – 1) x (Q – 1) = 1
  - Let us substitute the values of E, P, and Q in the equation.
  - We have (D x 5) mod (7 – 1) x (17 – 1) = 1.
  - That is, (D x 5) mod (6) x (16) = 1.
  - That is, (D x 5) mod (96) = 1
  - **After some calculations, let us take D = 77. Then the following is true: (77 x 5) mod (96) = 385 mod 96 = 1 which is what we wanted.**
- **For encryption, calculate the cipher text (CT) from the plain text (PT) as follows:**

  CT = PT^E mod N

  Let us assume that we want to encrypt plain text 10. Then, we have CT = 10^5 mod 119 = 100000 mod 119 = 40.
- Send CT as the cipher text to the receiver. Send 40 as the cipher text to the receiver.
- **For decryption, calculate the plain text (PT) from the cipher text (CT) as follows:**

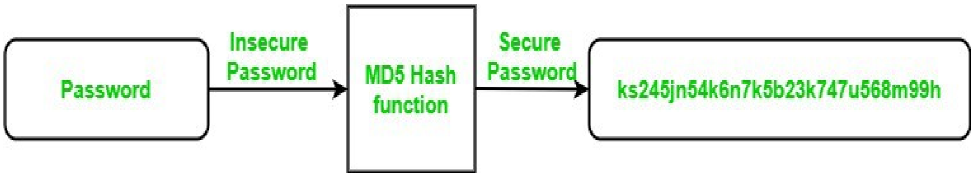
  PT = CT^D mod N It perform the following: PT = CT^D mod N

That is, PT = 40^{77}mod 119 = 10, which was the original plaintext.

Cryptographic Hash Functions

Hash Functions

- A hash function h maps a plaintext x to a fixed-length value x = h(P) called hash value or digest of P
  - Usually x is much smaller in size compared to P.
  - A collision is a pair of plaintexts P and Q that map to the same hash value, h(P) = h(Q)
  - Collisions are unavoidable
  - For efficiency, the computation of the hash function should take time proportional to the length of the input plaintext



Password

Insecure Password

MD5 Hash function

Secure Password

ks245jn54k6n7k5b23k747u568m99h

Simplex Example of Hash Functions

- Parity bit: map a binary bit stream to ‘1’ or ‘0’
  - Hash value space is only 2.
- Repeated addition of n-byte chunks without considering carry-on bits
  - Hash value space is 2^{8n}

Cryptographic Hash Functions

- A cryptographic hash function satisfies additional properties
  - Preimage resistance (aka one-way)
    - Given a hash value x, it is hard to find a plaintext P such that h(P) = x
  - Second preimage resistance (aka weak collision resistance)
    - Given a plaintext P, it is hard to find a plaintext Q such that h(Q) = h(P)
  - Collision resistance (aka strong collision resistance)
    - It is hard to find a pair of plaintexts P and Q such that h(Q) = h(P)
- Collision resistance implies second preimage resistance
- Hash values of at least 256 bits recommended to defend against brute-force attacks

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process. **(16-Bytes)**
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest **(20-Bytes)**
- There are many hash functions, but most of them do not satisfy cryptographic hash function requirements
  - **example: checksum**

Message-Digest Algorithm 5 (MD5)

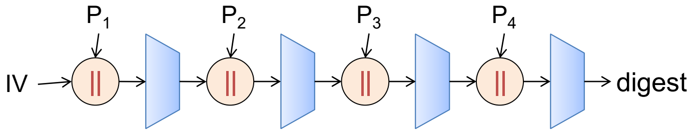
- Developed by Ron Rivest in 1991
- Uses 128-bit hash values
- Still widely used in legacy applications although considered insecure
- Various severe vulnerabilities discovered
- Chosen-prefix collisions attacks found by Marc Stevens, Arjen Lenstra and Benne de Weger
  - Start with two arbitrary plaintexts P and Q
  - One can compute suffixes S1 and S2 such that P||S1 and Q||S2 collide under MD5 by making 250 hash evaluations
  - Using this approach, a pair of different executable files or PDF documents with the same MD5 hash can be computed

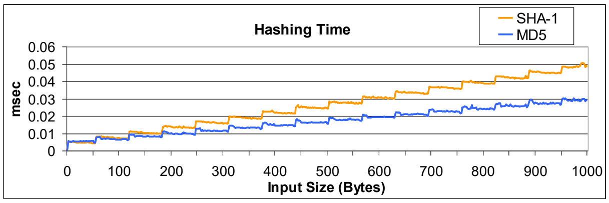
Secure Hash Algorithm (SHA)

- Developed by NSA and approved as a federal standard by NIST
- SHA-0 and SHA-1 (1993)
  - 160-bits
  - Considered insecure
  - Still found in legacy applications
  - Vulnerabilities less severe than those of MD5
- SHA-2 family (2002)
  - 256 bits (SHA-256) or 512 bits (SHA-512)
  - Still considered secure despite published attack techniques
- Public competition for SHA-3 announced in 2007

Iterated Hash Function

- A compression function works on input values of fixed length
- An iterated hash function extends a compression function to inputs of arbitrary length
  - padding, initialization vector, and chain of compression functions
  - inherits collision resistance of compression function
- MD5 and SHA are iterated hash functions





Birthday Attack

- The brute-force birthday attack aims at finding a collision for a hash function h
  - Randomly generate a sequence of plaintexts X_1, X_2, X_3,…
  - For each X_i compute $y_i = h(X_i)$ and test whether $y_i = y_j$ for some $j < i$
  - Stop as soon as a collision has been found
- If there are m possible hash values, the probability that the i-th plaintext does not collide with any of the previous i –1 plaintexts is $1 – (i – 1)/m$
- The probability F_k that the attack fails (no collisions) after k plaintexts is

$$F_k = (1 – 1/m)\ (1 – 2/m)\ (1 – 3/m) \ldots (1 – (k – 1)/m)$$

- Using the standard approximation $1 – x \approx e^{–x}$

$$F_k \approx e^{–(1/m + 2/m + 3/m + \ldots + (k–1)/m)} = e^{–k(k–1)/2m}$$

- The attack succeeds/fails with probability ½ when F_k = ½ , that is,

$$e^{–k(k–1)/2m} = ½$$

$$k \approx 1.17\ m^{½}$$

- We conclude that a hash function with b-bit values provides about b/2 bits of security