

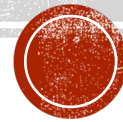
CRYPTOGRAPHY: INTRODUCTION

Introduction to Computer and Network Security

Silvio Ranise [silvio.ranise@unitn.it or ranise@fbk.eu]



UNIVERSITÀ
DI TRENTO



0

- Introduction to cryptography
 - A bit of history
 - Basic notions
 - The role of key management
 - Substitution and transposition
- Modern encryption techniques
 - Symmetric key encryption
 - Data Encryption Standard (DES)
 - Advanced Encryption Standard (AES)
 - Asymmetric key encryption
 - Trapdoor One-way functions
 - RSA
 - Diffie-Hellman

CONTENTS



1

HISTORY (1)

I

Those who are charged with the highest affairs know by experience how important it is to have a very trustworthy person with whom to reveal projects and decisions of the most secret nature, without ever having reasons to regret it.¹ This is rare, because men so often tend to perfidy, and so secure systems called *ciphers* have been devised, which would be useful, except for those who manage, through art and ingenuity, to read and interpret them. I admit that those in power may be served by those who are expert in such operations, if by their means the schemes and machinations of enemies can be revealed, but in my opinion, it is even more advantageous to be able to communicate our intentions to another, no matter how far away, in a way that no other mortal except the intended recipient of the missive is able to read them.

This present booklet of mine is a thorough treatment of both aspects of the question, paving the way for and directing the investigation into other people's secrets, as well as how, as we shall see, to protect secrets of your own.

The present circumstances persuaded me to send these notes of mine to you. Many friends who are devoted to you exhorted me to do so. I will be delighted if this work pleases you.¹

- Leon Battista Alberti. "*De componendis cifris*" (On writing in ciphers). **1466**
- Major turning point in the field of cryptography
- Many other work followed, each one proposing new technique to hide the content of messages to all except those who are supposed to receive them
 - Vigenère
 - Bellaso
 - Bernam
 - ...

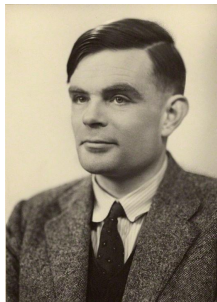
2

2

HISTORY (2)

- The *enigma* machine was used to secure communication of German military throughout the 2nd World War ...
- ... and it changed the course of human history

It is conjectured that the enigma machine shortened the war by 2 years saving around 14 million lives

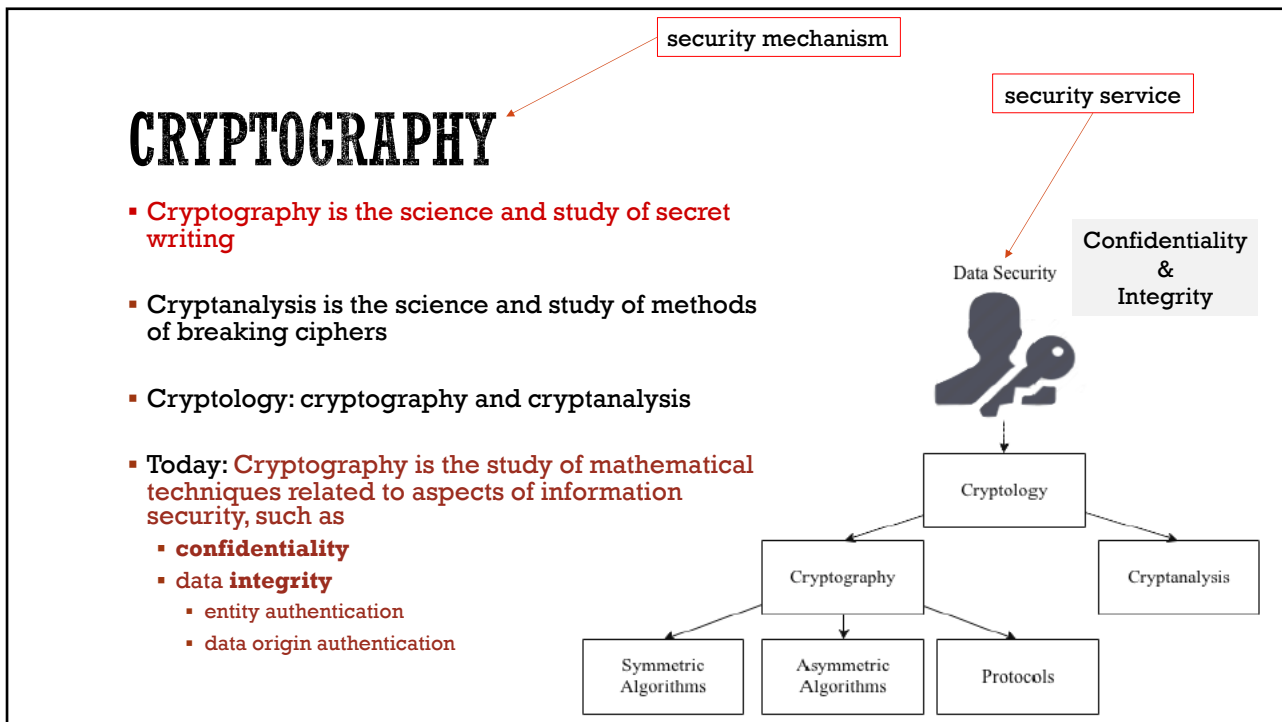


Alan Turing



3

3



4

CRYPTOGRAPHY RELOADED

- Cryptography is a security mechanism that includes a set of techniques for
 - scrambling or disguising data
 - so that it is available only to someone who can restore the data to its original form
- In current computer systems, cryptography provides a strong, economical basis for offering data security as a security service, i.e. a service for
 - keeping data secret (confidential) and
 - for verifying data integrity
- Crucial to cryptography is the notion of cryptosystem...



5

CRYPTOSYSTEM (1)

- A *cryptosystem* is a 5-tuple (E, D, M, K, C) where
 - E is an *encryption* algorithm
 - D is a *decryption* algorithm
 - M is the set of *plaintexts*
 - K is the set of *keys*
 - C is the set of *ciphertexts*
- Abstractly, E and D can be characterized as functions:
 - $E : M \times K \rightarrow C$
 - $D : C \times K \rightarrow M$

$$D(E(m, k), k) = m$$

Basic Encryption & Decryption



6

6

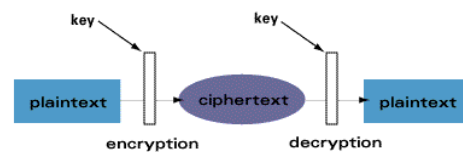
CRYPTOSYSTEM (2)

- A *cryptosystem* is a 5-tuple (E, D, M, K, C) where
 - E is an *encryption* algorithm
 - D is a *decryption* algorithm
 - M is the set of *plaintexts*
 - K is the set of *keys*
 - C is the set of *ciphertexts*
- Abstractly, E and D can be characterized as functions:
 - $E : M \times K \rightarrow C$
 - $D : C \times K \rightarrow M$

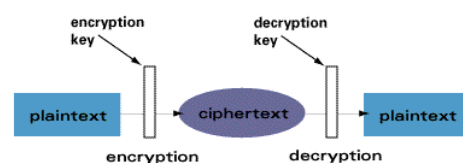
$$D(E(m, k), k) = m$$

encryption and decryption keys
are not necessarily the same

Symmetric Algorithms



Asymmetric Algorithms

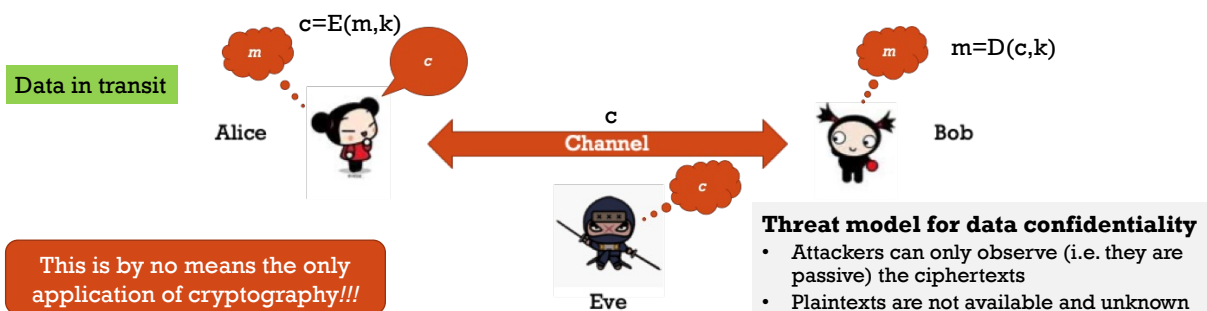


7

7

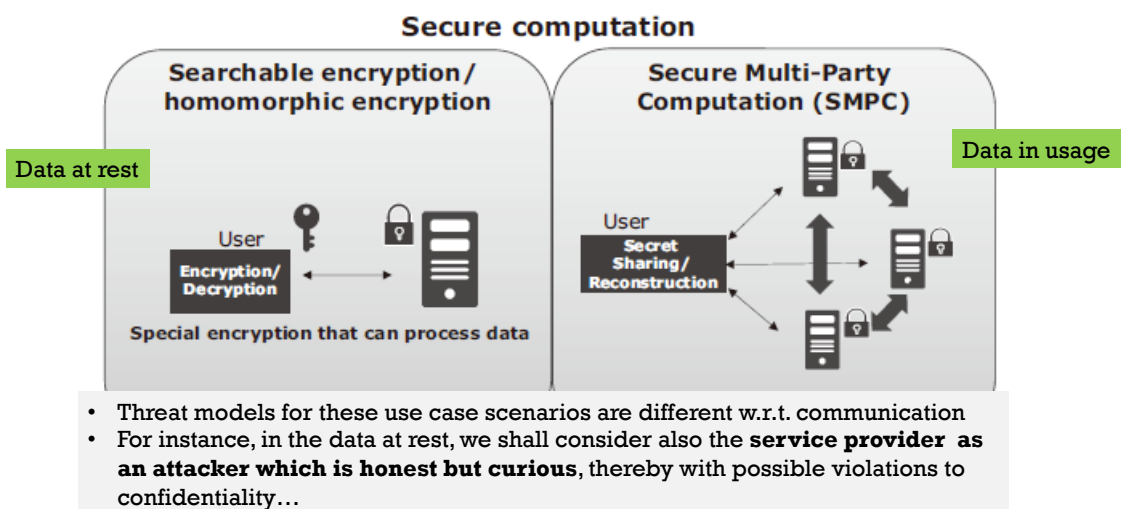
APPLICATION: COMMUNICATION SECURITY

- Security services provided by cryptographic mechanisms
 - Data confidentiality**: encryption algorithms hide the content of messages;
 - Data integrity**: integrity check (e.g., hash) functions provide the means to detect whether a document has been changed;
 - Data origin authentication**: message authentication codes (e.g., HMAC) or digital signature algorithms provide the means to verify the source and integrity of a message



8

OTHER APPLICATIONS OF CRYPTOGRAPHY



9

CRYPTOSYSTEM (3)

- A *cryptosystem* is a 5-tuple (E, D, M, K, C) where
 - E is an *encryption* algorithm
 - D is a *decryption* algorithm
 - M is the set of *plaintexts*
 - K is the set of *keys*
 - C is the set of *ciphertexts*
- Abstractly, E and D can be characterized as functions:
 - $E: M \times K \rightarrow C$
 - $D: C \times K \rightarrow M$

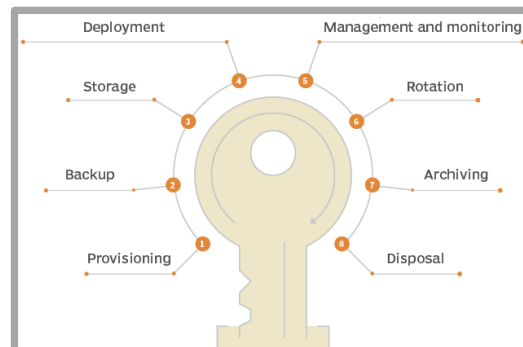
such that $D(E(m, k), k) = m$ and one or both of

the following properties are satisfied

- **confidentiality.** If someone knows $E(m, k)$ but not k , then m cannot be computed
- **integrity.** If one chooses c but does not know k , then it is impossible to compute $D(c, k)$

- **provisioning** or generating keys for organizational needs;
- **backing up** keys that must be recoverable after system failures
- **storing** keys securely
- **deploying** keys to systems that need them
- **managing and monitoring** use of deployed keys
- **rotating** keys to prevent overexposing them to sensitive material
- **archiving** keys that are no longer being actively used
- **disposing** keys that are no longer required.

- **Kerckhoffs' principle:** do not rely on the secrecy of algorithms; the **key should be the only secret that needs protection**



10

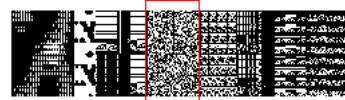
10

ABOUT KEYS

- A key is an input to a cryptographic algorithm used to obtain confidentiality, integrity, authenticity or other property over some data
 - The security of the cryptosystem often **depends on keeping the key secret** to some parties
 - The *key space* is the set of all possible keys
 - *Entropy* is a measure of the variance in keys
 - typically measured in bits
- Keys are often stored in some secure place:
 - passwords, on disk keyrings, ...
 - Trusted Platform Modules (TMP), secure smartcards, ...
- ... and sometimes not, e.g., certificates

- For most cryptographic systems the cryptographic **keys should be as random as possible**
- In other words, the keys should exhibit high **entropy**, i.e. the **randomness** collected by an operating system or application for use in cryptography
- Intuitively, randomness is the apparent or actual lack of pattern or predictability in events

bitmap representations of different keys... why the key highlighted is "better" than the others?



11

11

ABOUT KEY DISTRIBUTION

- The two parties to an exchange must share the same key, and that key must be protected from access by others
- Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key
- **The strength of any cryptographic system rests with the key distribution technique**
- Example approaches for entities A and B:
 1. A key could be selected by A and physically delivered to B
 2. A third party could select the key and physically deliver it to A and B
 3. If A and B have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key
 4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B

12

12

ABOUT “COMPUTATIONALLY SECURE”

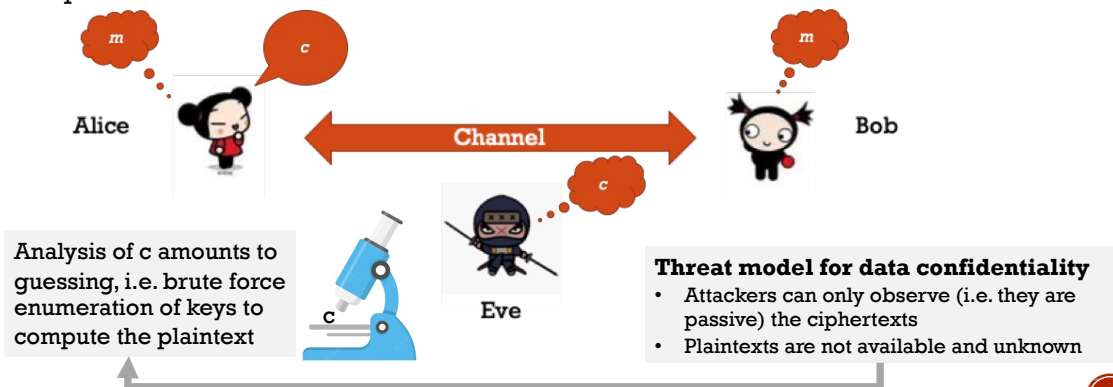
- An encryption scheme is **computationally secure** if the ciphertext generated by the scheme meets one or both of the following criteria:
 - The cost of breaking the cipher exceeds the value of the encrypted information
 - The time required to break the cipher exceeds the useful lifetime of the information
- Very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully
- Assuming there are no inherent mathematical weaknesses in the algorithm, then a **brute-force approach** is indicated, and so one can make some reasonable estimates about costs and time
- **Brute-force approach** = trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained
 - *On average, half of all possible keys must be tried to achieve success*

13

13

“COMPUTATIONALLY SECURE” IN COMMUNICATION

- The task of the attacker is very difficult and a lot of computational power is required to mount such an attack



14

“COMPUTATIONALLY SECURE” IN SHORT

- $E(k, P) = C$
 - Computing C from P is hard without k , computing C from P with k is easy
- $D(k, C) = P$
 - Computing P from C is hard, computing P from C with k is easy
- P : plaintext
- C : ciphertext
- k : key
- A **trapdoor one-way function** is a one-way function with an additional requirement
- Informally, a **one-way function** might be described as a function for which evaluation in one direction is straightforward, while computation in the reverse direction is far more difficult.
- Such a function becomes a trapdoor one-way function when we add the requirement that **computation in the reverse direction becomes straightforward when some additional (trapdoor) information is revealed**

15

HASH VS ENCRYPTION FUNCTIONS (1)

- A **hash** function is a **one-way function**
 - a function for which evaluation in one direction is straightforward, while computation in the reverse direction is far more difficult
- An **encryption** function is a **trapdoor one-way function**
 - a one-way function such that computation in the reverse direction becomes straightforward when the key is revealed
- Hash functions shall be used when there is no need to recover the input from the output whereas encryption functions shall be used when there is a need to efficiently recover the input from the output provided that a key is available

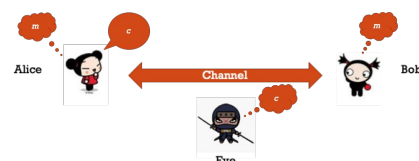


16

16

HASH VS ENCRYPTION FUNCTIONS (2)

- Hash functions shall be used when there is no need to recover the input from the output whereas encryption functions shall be used when there is a need to efficiently recover the input from the output provided that a key is available
- **Hash function use case scenario: integrity**
 - compute the digest of a message and send along with the message
 - on the receiving side, digest can be recomputed using the same hash and result compared with the digest sent with the message
 - if equal, then message has not been tampered with
 - otherwise, there is a violation of message integrity
 - confidentiality is not guaranteed as the message is sent in plaintext

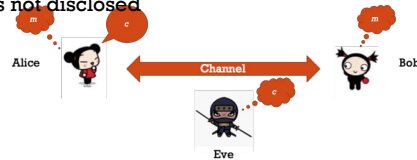


17

17

HASH VS ENCRYPTION FUNCTIONS (3)

- Hash functions shall be used when there is no need to recover the input from the output whereas encryption functions shall be used when there is a need to efficiently recover the input from the output provided that a key is available
- **Encryption function use case scenario: confidentiality**
 - assume sender and receiver share the same secret key
 - compute the ciphertext of a plaintext message with available key
 - send ciphertext
 - on the receiving side, ciphertext can be decrypted with the shared secret key
 - secrecy of message is preserved provided secret key is not disclosed
 - integrity is not guaranteed: if an attacker flip one bit of the ciphertext, the receiver cannot detect it



18

18

CRYPTOGRAPHY IS NO SILVER BULLET

- Cryptographic keys are sensitive data stored in a computer system
 - **Access control mechanisms in the computer system have to protect these keys**
- Cryptography (alone) is rarely ever the solution to a security problem
- Cryptography is a translation mechanism
 - usually *converting a communication security problem into a key management problem*
 - ultimately into a computer security problem or into the security of a cryptographic protocol for key distribution

19

19

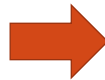
THE ESSENCE OF CRYPTO: ENCRYPTION

- Algorithm used to make content unreadable by all but the intended receivers

- $E(\text{key}, \text{plaintext}) = \text{ciphertext}$

- $D(\text{key}, \text{ciphertext}) = \text{plaintext}$

- **Algorithm is public, key is private**



This is opposite to the principle of **security by obscurity**, i.e. reliance on design or implementation secrecy as the main method of providing security to a system or component.

- Block vs. Stream Ciphers
 - Block: input is fixed blocks of same length
 - Stream: stream of bits

20

20

A CLOSER LOOK AT ENCRYPTION

- Two types of transformations
 - **Substitution**: each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element
 - In other words, **letters are replaced by other letters**
 - **Transposition**: elements in the plaintext are rearranged
 - In other words, **same letters but arranged in a different order**
- Fundamental requirement: no information shall be lost (i.e., all operations be reversible)
- Most encryption systems use multiple stages of substitutions and transpositions

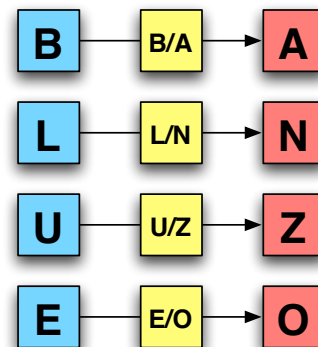
We describe transformations over letters as they are more intuitive...
however, the same ideas applies to strings of bits

21

21

SUBSTITUTION CIPHERS

- In brief
 - Substitutes one symbol for another
 - The key is the substitution
- Longer explanation
 - A **substitution cipher** is a method of encrypting by which units of plaintext are replaced with ciphertext, according to a fixed system
 - The "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth
 - The receiver deciphers the text by performing the inverse substitution

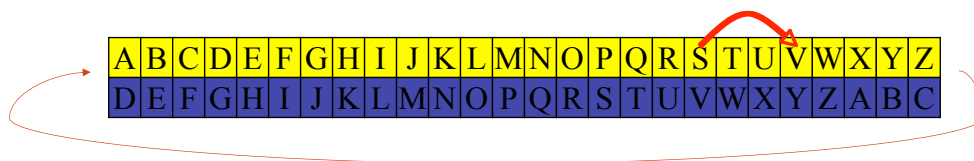


22

22

CAESAR CIPHER

- Substitution cipher
- Every character is replaced with the character, e.g., three slots to the right



- The key is the **number of characters to shift the cipher alphabet**
 - In the case above three

23

23

CAESAR CIPHER

- Substitution cipher
- Every character is replaced with the character, e.g., three slots to the right



- The key is the **number of characters to shift the cipher alphabet**
 - In the case above three

Also called
ROT_k
for ROTATE by k

24

24

ROT_k: MATHEMATICALLY

- Translate all of our characters to numbers:
 - 'a' -> 0 'b' -> 1 'c' -> 2, ..., 'z' -> 25
- Represent the ROT_k encryption function, $e(x)$, where x is the character we are encrypting, as:

$$e(x) = (x + k) \pmod{26}$$

where k is the key (the shift) applied to each letter

- After applying this function the result is a number which must then be translated back into a letter
- Decryption function

$$d(x) = (x - k) \pmod{26}$$

25

25

TRY TO DECRYPT THIS

“orgursbeprijvgulbh”

What is a key in
Caesar cipher?

Hint: try a brute-force approach

1. Consider a new key
2. Apply decryption
3. Is the obtained cleartext meaningful?
4. If not, go back to step 1

26

26

CRYPTANALYSIS OF ROT_k

- ROT_k cipher is probably the easiest of all ciphers to break
- Since the shift has to be a number between 1 and 25 (0 or 26 would result in an unchanged plaintext), one can simply try each possibility and see which one results in a piece of readable text
 - Brute-force attack
- More principled approach
 - Calculate the frequency distribution of the letters in the cipher text, i.e. count how many times each letter appears
 - **English text has a very distinct distribution that can be used help crack codes**
 - 'e' is the most common letter (appears almost 13% of times)
 - 'z' is the least frequent letter (appears only 1% of times)
 - Find the shift (i.e. the key) that causes the ciphertext frequencies to match up closely with the natural English frequencies, then decrypt the text using that shift (key)

27

27

VIGENÈRE CIPHER

1st column

- A associated to 0
- L associated to 11
- $0+11 \bmod 26 = 11$ associated to L

2nd column

- T associated to 19
- E associated to 4
- $19+4 \bmod 26 = 23$ associated to X

- It can be seen as a generalization of the Caesar cipher whereby several Caesar ciphers in sequence with different shift values are used
- As for the Caesar cipher, assume to associate the letters a, b, c, ..., z with the numbers 0, 1, 2, ..., 25
 - First, select a keyword
 - Then, write out the keyword repeatedly underneath the plaintext until every plaintext letter has a keyword letter beneath it
 - Finally, encrypt each plaintext letter using a Caesar Cipher, whose key is the number associated with the keyword letter written beneath it
- Example

A	T	T	A	C	K	A	T	D	A	W	N	plaintext
L	E	M	O	N	L	E	M	O	N	L	E	keyword repeated to match plaintext length
L	X	F	O	P	V	E	F	R	N	H	R	ciphertext

28

28

VIGENÈRE CIPHER

A	T	T	A	C	K	A	T	D	A	W	N
L	E	M	O	N	L	E	M	O	N	L	E
L	X	F	O	P	V	E	F	R	N	H	R

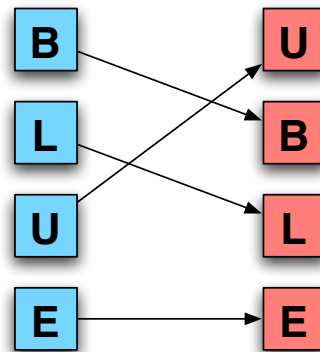
- Notice that the same letter A is encrypted with 4 different letters, namely L, O, E, and N depending on the position of A in the plaintext
- Notice that letter T occurs 3 times but it is mapped to 2 different letters
- This makes frequency analysis more difficult on this cipher than with Caesar cipher
- Since the key is the keyword, its length l determines the **size of the key space**, namely l^{26}
- For increasing length l of the keyword, brute forcing becomes increasingly complex until it becomes impossible...
- ... **when the length of the keyword is the same as the length of the plaintext**, a separate Caesar Cipher is used to encrypt each plaintext letter, which makes it impossible to determine the correct plaintext without the key
 - In this case, the **Vigenère cipher cannot be broken**, if additionally the key is used only once

29

29

TRANSPOSITION CIPHERS

- In brief
 - Scrambles the symbols to produce output
 - The key is the **permutation** of symbols
- Longer explanation
 - In a transposition cipher, **the units of the plaintext are rearranged in a different and usually quite complex order**, but the units themselves are left unchanged
 - Several variants possible
- Comparison with substitution ciphers
 - In a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered



30

30

COLUMNAR CIPHER

- In a columnar transposition, the **message is written out in rows of a fixed length, and then read out again column by column**, and the columns are chosen in some scrambled order
- Both **the width of the rows and the permutation of the columns are usually defined by a keyword**
- For example, the keyword ZEBRAS is of length 6 (so the rows are 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5"

Example

Plain text: WE ARE DISCOVERED. FLEE AT ONCE

Keyword: Z E B R A S
(length 6)

6 3 2 4 1 5

Toward the ciphertext using
a grid (with 6 columns)

6	3	2	4	1	5
W	E	A	R	E	D
I	S	C	O	V	E
R	E	D	F	L	E
E	A	T	Q	N	C
E	Q	K	J	E	U

Ciphertext:

EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE

Nulls for filling up the last row to the exact length of 6 (not part of the message, can be randomly selected)

31

31

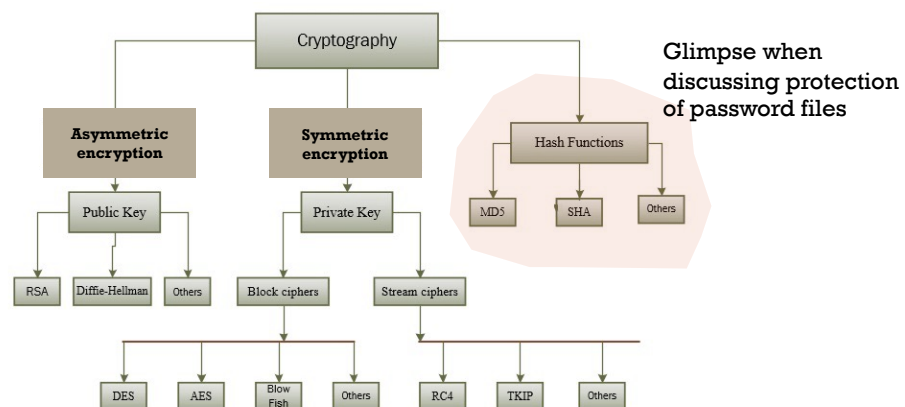
FROM CLASSIC TO MODERN CRYPTO

- With the quick development of the computer, it become difficult to depend on the elementary classical technique as it is easy to decrypt the ciphertext
 - **Encryption technique** enters
 - new process depends on the algorithms, which **use the bits instead of characters**, so
 - Stream encryption
 - Block encryption
 - Public-key encryption
- Symmetric key encryption
- Asymmetric key encryption

32

32

OVERVIEW OF MODERN CRYPTOGRAPHY



33

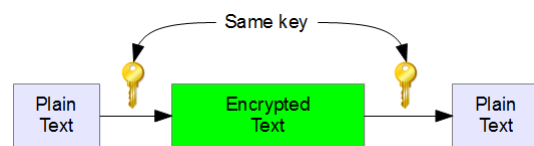
33

SYMMETRIC KEY CRYPTOGRAPHY

- Symmetric keys, where a single key (k) is used is used for **E** and **D**

$$D(k, E(k, p)) = p$$

- All (intended) receivers have access to key



- Note: **Management of keys determines who has access to encrypted data**

34

34

SYMMETRIC KEY CRYPTO: TYPES

- **Stream ciphers**: encrypt sequences of “short” data blocks under a changing key stream
 - Security relies on design of key stream generator
 - Encryption can be quite simple, e.g., XOR
 - Typical block length: 1 bit/byte
- **Block ciphers**: encrypt sequences of “long” data blocks without changing the key
 - Security relies on design of encryption function
 - Typical block length: 64/128 bits

35

35

SOME REMARKS ON SYMMETRIC CRYPTOGRAPHY

- Stream ciphers
 - They use the idea underlying the Vigenère cipher adapted to stream of bits rather than texts of letters
- Block ciphers
 - They use substitution and transposition adapted to manipulate blocks of bits rather than letters in texts
 - Substitutions are called **S-boxes** and aim to “hide” the relationship between the key and the ciphertext, i.e. to obtain confusion
 - **Confusion** means that each bit of the ciphertext should depend on several parts of the key
 - Transpositions are called **P-boxes** and aim to permute bits across S-boxes inputs, i.e. to obtain diffusion
 - **Diffusion** means that if we change a single bit of the plaintext, then about half of the bits in the ciphertext should change, this is similar to the avalanche effect that we have discussed in conjunction with hash functions

36

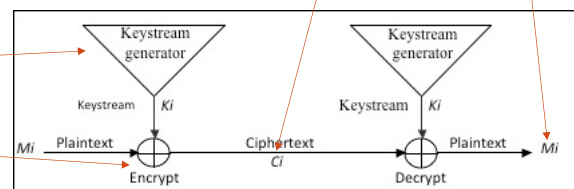
36

STREAM CIPHERS (1)

- Operates on streams of plaintext and ciphertext one bit at a time
- The same plaintext bit will encrypt to a different bit, every time it is encrypted
- Stream ciphers can be designed to be exceptionally fast, much faster than any block cipher in hardware, and have less complex hardware circuitry

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

Generates a sequence of (pseudo) random bits



simplest implementation of a stream cipher

37

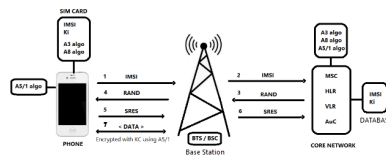
37

STREAM CIPHERS (2)

- Typically used in real time applications such as mobile communications and video streaming

- Examples

- GSM traffic encryption A5/1



<https://www.blackhillsinfosec.com/gsm-traffic-and-encryption-a5-1-stream-cipher/>

- E0



- RC4



38

38

BLOCK CIPHERS

- A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length
 - Typical size of the block: between 64 and 256 bits
- A block cipher breaks the message (M) into successive blocks $M_1, M_2, M_3, \dots, M_n$, and enciphers each M_i with the same key k
- Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are well known examples of block cipher systems

39

39

SYMMETRIC KEY CRYPTO: AN EXAMPLE

- **Data Encryption Standard (DES)**
 - Designed by IBM in the 1970s and adopted by NIST in 1977 for commercial and unclassified government applications
 - **Feistel block-cipher** employing a 56-bit key that operates on 64-bit blocks
 - Designed specifically to yield fast hardware implementations and slow software implementations, although this latter point is not significant today since the speed of computer processors is several orders of magnitude faster
 - The NSA also proposed a number of tweaks to DES that many thought were introduced in order to weaken the cipher, but analysis in the 1990s showed that the NSA suggestions actually strengthened DES
- DES was defined in American National Standard X3.92 and three Federal Information Processing Standards (FIPS), all withdrawn in 2005

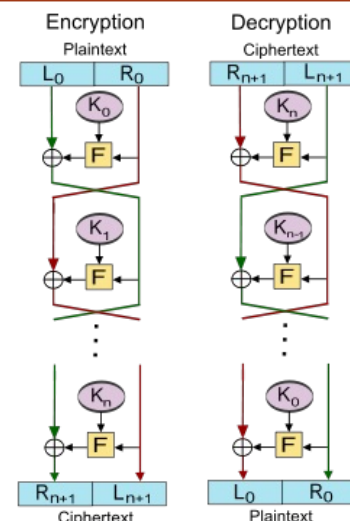
40

40

FEISTEL (1973)

- It is a **block cipher**, i.e. a **deterministic algorithm** operating on fixed-length groups of **bits**, called a **block**, with an unvarying transformation that is specified by a **symmetric key**
- How it works
 - The block of plain text to be encrypted is split into two equal-sized halves (L,R)
 - The round function F (**substitution**) is applied to one half, using a subkey K. Then the output is **XORed** with the other half
 - The two halves are then swapped
- DES has $n = 16$ rounds
 1. Split input 64 bits into R and L parts of 32 bits each
 2. Compute $X = F(R, K)$
 3. Compute $L \text{ xor } X$
 4. Let the new 32 bits R be X
 5. Let the new 32 bits L be R

- Instead of using the same key in each round, a different subkey is derived from the main key
- When decrypting, subkeys must be used in reverse order



41

41

CRACKING DES: THE ELECTRONIC FRONTIER FOUNDATION (EFF) EFFORT (CIRCA 1998)

- The easiest known way to build a practical DES Cracker is to have it try every key until it finds the right one
- EFF DES Cracker consists of an ordinary personal computer with a large array of custom "Deep-Crack" chips
- Software parallelize searching for a DES key
 - A single DES-Cracker chip could find a key by searching for many years
 - A thousand DES-Cracker chips can solve the same problem in one thousandth of the time
 - The actual machine EFF built contains about 1,500 chips
- **It took the EFF DES Cracker <3 days to find a 56-bit key by searching a total of 17,902,806,669,197,312 keys (an average of 88 billion keys per second)**
- **Cost of the project: < \$250,000** (\$80,000 for design, integration, and test; \$130,000 for hw)
- SW developed in 4-5 weeks; entire project completed within about eighteen months

42

42

Lesson learnt from DES cracking...
make encryption parametric in the key size!

AFTER DES: AES

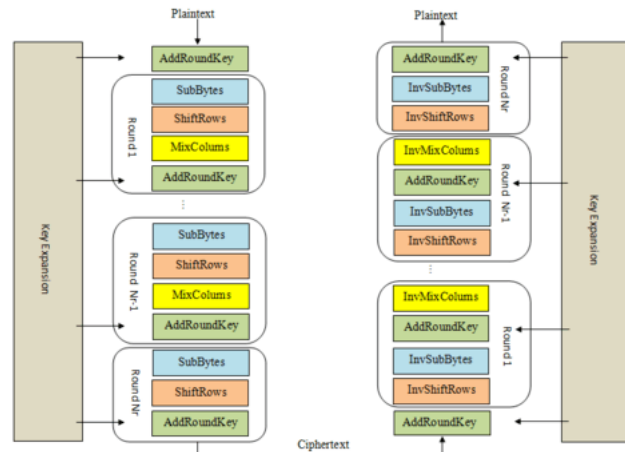
- In 1997, NIST initiated a public, 4-1/2 year process to develop a new secure cryptosystem for U.S. government applications (as opposed to the closed process in the adoption of DES 25 years earlier)
- Result was the Advanced Encryption Standard (**AES**) that became the official successor to DES in December 2001
- AES uses a symmetric key crypto scheme called Rijndael, a block cipher designed by Belgian cryptographers Joan Daemen and Vincent Rijmen
 - The algorithm can use a **variable** block length and **key length**; the latest specification allowed any combination of keys lengths of **128, 192, or 256 bits** and blocks of length 128, 192, or 256 bits
- NIST initially selected Rijndael in October 2000 and formal adoption as the AES standard came in December 2001

43

43

AES, OVERVIEW

- The encryption process is based on a series of table lookups and XOR operations which are **very fast operations to perform** on a computer
- Decryption of a ciphertext consists of conducting the encryption process in the reverse order
- However, unlike for a Feistel Cipher, the encryption and decryption algorithms do have to be separately implemented, although they are very closely related



44

44

ABOUT AES (2001)

- AES is the cryptographic algorithm for use by U.S. government organizations to protect sensitive, unclassified information
 - Commercial and other non-federal organizations are invited-but not required-to adopt and implement DES
- No one can be sure how long the AES-or any other cryptographic algorithm-will remain secure. However, NIST's DES was a U.S. government standard for approximately 20 years before it became practical to mount a brute force attack with specialized hardware. The AES supports significantly larger key sizes than what DES supports. Barring any attacks against AES that are faster than key exhaustion, and even with future advances in technology, AES has the potential to remain secure well beyond 20 years
 - In AES, there are approximately: 3.4×10^{38} possible 128-bit keys; 6.2×10^{57} possible 192-bit keys; and 1.1×10^{77} possible 256-bit keys.
 - In DES, there are approximately 7.2×10^{16} possible DES keys
 - There are on the order of **10^{21} times more AES 128-bit keys than DES 56-bit keys**

45

45

ASYMMETRIC KEY CRYPTOGRAPHY

- Also called **Public Key Cryptography (PKC)**
- The most significant development in cryptography in the last 300-400 year
- First described publicly by Stanford University Prof. Martin Hellman and graduate student Whitfield Diffie in **1976**
- Two-key crypto system in which two parties could engage in a **secure communication over a non-secure communications channel without sharing a secret key**

46

46

PKC: ONE-WAY FUNCTIONS

- Main ingredient of PKC: **existence of so-called one-way functions**, i.e. mathematical functions that are *easy to compute whereas their inverse function is relatively difficult to compute*

To be picky, we should be able to quickly invert the function if some additional information becomes known, e.g., a suitable key. We call this kind of functions **trapdoor one-way functions**

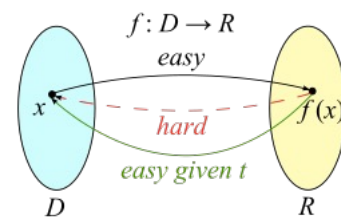
- Examples
 - *Multiplication vs. factorization*
 - Given prime numbers, say 3 and 7, it is easy to calculate the product
 - Given number 21 that is a product of two primes, it is not easy to determine the prime factors
 - Problem becomes much harder if we start with primes that have, say, 400 digits or so, because the product will have ~800 digits
 - *Exponentiation vs. logarithms*
 - Take the number 3 to the 6th power; it is relatively easy to calculate $3^6 = 729$
 - Start with number 729; it is difficult to determine the two integers, x and y s.t. $\log_x 729 = y$
 - **With suitable parameters**, these problems are a basis for many cryptographic algorithms
 - However, **not all instances of these problems are difficult to solve**

47

47

ON TRAPDOOR ONE WAY FUNCTIONS

- **Example**
 - 6,895,601 is the product of two prime numbers
 - What are those numbers?
 - A brute force solution is to try dividing 6,895,601 by several prime numbers until finding the answer
 - If one is told that 1,931 is one of the numbers, one can find the answer by computing
 - $6,895,601 \div 1,931 = 3,571$
 - While computers can guess all of the possible answers within a second, by considering larger and larger primes, inverting similar functions becomes more and more computationally expensive
 - Indeed, the problem is **believed** to be difficult from a computational point of view
 - This means that there is no mathematical proof that it is indeed computationally hard

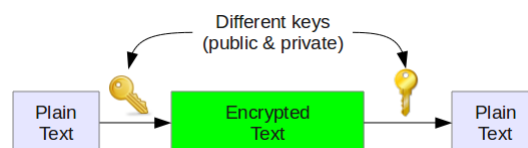


48

48

PKC: BASIC IDEA

- Use two keys: one to encode and the other to decode such that they are...



- ... *mathematically related* **although knowledge of one key does not allow someone to easily determine the other key**
- It **does not matter which key is applied first**, but that both keys are required for the process to work

49

49

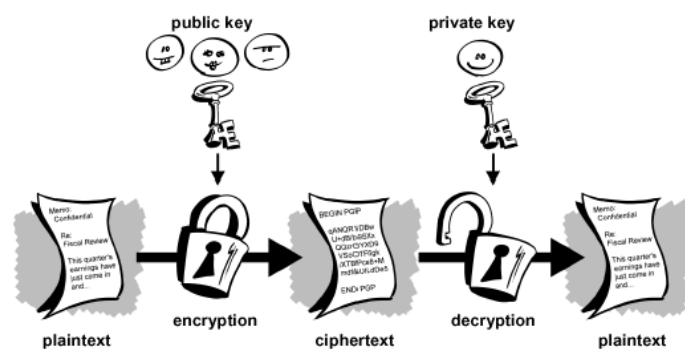
PKC: BASIC IDEA (CONT'D)

- One of the keys is designated the *public key* and may be advertised as widely as the owner wants
- The other key is designated the *private key* and is never revealed to another party
- How to send a message under this scheme: suppose Alice wants to send Bob a message
 - Alice encrypts some information using Bob's public key
 - Bob decrypts the ciphertext using his private key
- Method could be also used to prove who sent a message:
 - Alice could encrypt some plaintext with her private key
 - When Bob decrypts using Alice's public key, he knows that Alice sent the message (*authentication*) and Alice cannot deny having sent the message (*non-repudiation*)

50

50

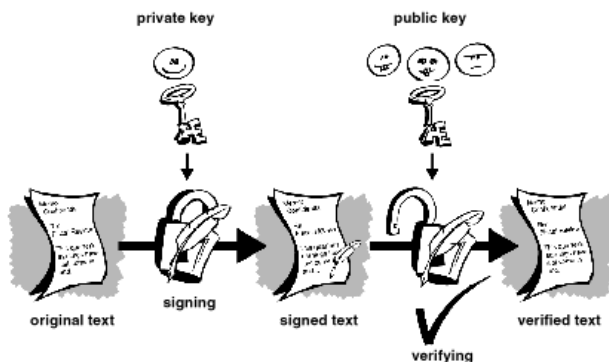
PKC FOR DATA CONFIDENTIALITY



51

51

PKC AND SIGNATURES



- Only sender could have signed
- Like handwritten signature... only more difficult to fake

A **digital signature** gives a recipient reason to believe that

- a message was created by a known sender (**authentication**),
- the sender cannot deny having sent the message (**non-repudiation**), and
- the message was not altered in transit (**integrity**)

52

52

PKC ALGORITHMS USED TODAY

- **RSA (Rivest, Shamir, Adleman)**
 - Used in hundreds of software products and can be used for **key exchange, digital signatures, or encryption** of small blocks of data
 - Mathematical "trick" of RSA: **relatively easy to compute products compared to computing factorizations**
 - RSA uses a variable size encryption block and a variable size key
 - Key-pair is derived from a very large number, n , that is the product of two prime numbers chosen according to special rules
 - Primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors
 - **An attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure**
 - The ability for computers to factor large numbers, and therefore attack RSA scheme, is rapidly improving and systems today can find the prime factors of numbers with more than 200 digits
 - Nevertheless, if a large number is created from two prime factors that are roughly the same size, there is no known factorization algorithm that will solve the problem in a reasonable amount of time; a 2005 test to factor a 200-digit number took 1.5 years and over 50 years of compute time
- **DH (Diffie-Hellman)**
 - After the RSA algorithm was published, Diffie and Hellman came up with their own algorithm
 - DH is **used for secret-key key exchange only**, not for authentication or digital signatures

53

53

PKC ALGORITHMS USED TODAY

- **RSA (Rivest, Shamir, Adleman)**
 - Used in hundreds of software products and can be used for **key exchange, digital signatures, or encryption** of small blocks of data
 - Mathematical "trick" of RSA: **relatively easy to compute product compared to computing factorizations**
 - RSA uses a variable size encryption block and a variable size key
 - Key-pair is derived from a very large number, n , that is the product of two prime numbers chosen according to special rules
 - Primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors
 - **An attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure**
 - The ability for computers to factor large numbers, and therefore attack RSA scheme, is rapidly improving and systems today can find the prime factors of numbers with more than 200 digits
 - Nevertheless, if a large number is created from two prime factors that are roughly the same size, there is no known factorization algorithm that will solve the problem in a reasonable amount of time; a 2005 test to factor a 200-digit number took 1.5 years and over 50 years of compute time
- **DH (Diffie-Hellman)**
 - After the RSA algorithm was published, Diffie and Hellman came up with their own algorithm
 - DH is **used for secret-key key exchange only**, not for authentication or digital signatures

54

54

ON PKC AND FACTORISATION

addition	multiplication	factoring
Easy		Difficult
$\begin{array}{r} 123 \\ + 654 \\ \hline 777 \end{array}$	$\begin{array}{r} 123 \\ \times 654 \\ \hline 492 \\ 615 \\ 738 \\ \hline 80442 \end{array}$	$\begin{array}{l} 221 = ? \times ? \\ 221/2 = \\ 221/3 = \\ 221/5 = \\ 221/7 = \\ 221/11 = \\ 221/13 = \\ 221 = 13 \times 17 \end{array}$

55

55

ON PKC AND FACTORISATION

addition	multiplication	factoring
Easy		Difficult
		$221 = ? \times ?$ $221/2 =$ $221/3 =$ $221/5 =$ $221/7 =$ $221/11 =$ $221/13 =$ $221 = 13 \times 17$

- A **prime number** (or a **prime**) is a natural number greater than 1 that has no positive divisors other than 1 and itself
- Fundamental theorem of arithmetic: any integer greater than 1 is either a prime itself or can be expressed as a product of primes that is unique up to ordering
- Slow method of verifying primality of a number n : trial division
 - Testing whether n is a multiple of any integer between 2 and \sqrt{n}
 - Why? If n is not prime, then there exist a and b s.t. $n = a*b$ and so, obviously, a or b can be at most \sqrt{n}
 - Improvement: only prime numbers should be considered

56

56

ON PKC AND FACTORIZATION (CONT'D)

- Let p and q two large **prime** numbers
- Let $N = p*q$ be the **modulus**
- Choose e relatively prime to $(p-1)*(q-1)$
- Find d s.t. $e*d = 1 \bmod (p-1)(q-1)$
- Set **public key** to (N, e)
- Set **private key** to d
- To **encrypt** plaintext $0 < M < N$, compute
 - $C = M^e \bmod N$
- To **decrypt** ciphertext C , compute
 - $M = C^d \bmod N$
- Notice that** $C^d \bmod N = (M^e \bmod N)^d \bmod N = M^{e*d} \bmod N = M \bmod N = M$
- Since N and e are public...
- ... if attacker can factor N , he/she can use it to find d since $e*d = 1 \bmod (p-1)(q-1)$

d is the inverse
of e modulo

- For a positive integer n , two integers a and b are said to be *congruent modulo n* , if their difference $a - b$ is an integer multiple of n (i.e., if there is an integer k such that $a - b = kn$)
- In symbols, $a = b \bmod n$
- Ex: $38 = 14 \bmod 12$ since $38 - 14 = 2*12$
- Two integers a and b are said to be **relatively prime** or **coprime** if the only positive integer (factor) that divides both of them is 1

Modular arithmetic and exponentiation

- Applying exponents in modular arithmetic can be done before or after simplifying:

$$a^k \bmod n = (a \bmod n)^k$$
- Ex:
 - $6^2 \bmod 4 = 36 \bmod 4 = 0$
 - $6^2 \bmod 4 = (6 \bmod 4)^2 \bmod 4 = (2)^2 \bmod 4 = 4 \bmod 4 = 0$

57

57

EXAMPLE

- Pick prime numbers $p=11$ and $q=3$
- Compute $N=p*q=11*3=33$
- Pick $e=3$ as it is relatively prime with both $p-1=10$ and $q-1=2$
- Compute d such that $e*d \equiv 1 \pmod{(p-1)*(q-1)}$
 - $3*d \equiv 1 \pmod{10*2}$
 - $3*d \equiv 1 \pmod{20}$
 - $3*d \equiv 1 + 20*k$ for some k
 - Find d such that $3*d-1$ divides 20
 - By testing $d=1, 2, \dots$, it is possible to find $d=7$ as $3*7-1=20$
- Set **public** key to $(N,e) = (33,3)$
- Set **private** key to $d=7$

Recall that

"multiplication of two large primes"
is *believed* to be a trapdoor one-way function

This is so because none has come up with a formal proof but repeated attempts of cryptographers failed to prove the contrary...

$a \equiv b \pmod{c}$ iff
there exists k such that
 $a = b + c*k$
or, equivalently,
 $a-b$ divides c

58

58

EXAMPLE (CONT'D)

- Consider plaintext message $M=7$
- Compute the ciphertext $C = M^e \pmod{N} = 7^3 \pmod{33} = 343 \pmod{33} = 13$
- Check decryption: $M = C^d \pmod{N} = 13^7 \pmod{33} = 62748517 \pmod{33} = 7$

It is possible to avoid the computation of 13^7 by exploiting the following equality:
 $a \equiv b*c \pmod{n} \iff (b \pmod{n}) * (c \pmod{n}) \pmod{n}$

59

59



RON RIVEST, ADI SHAMIR & LEN ADLEMAN

acm
AWARD
2002

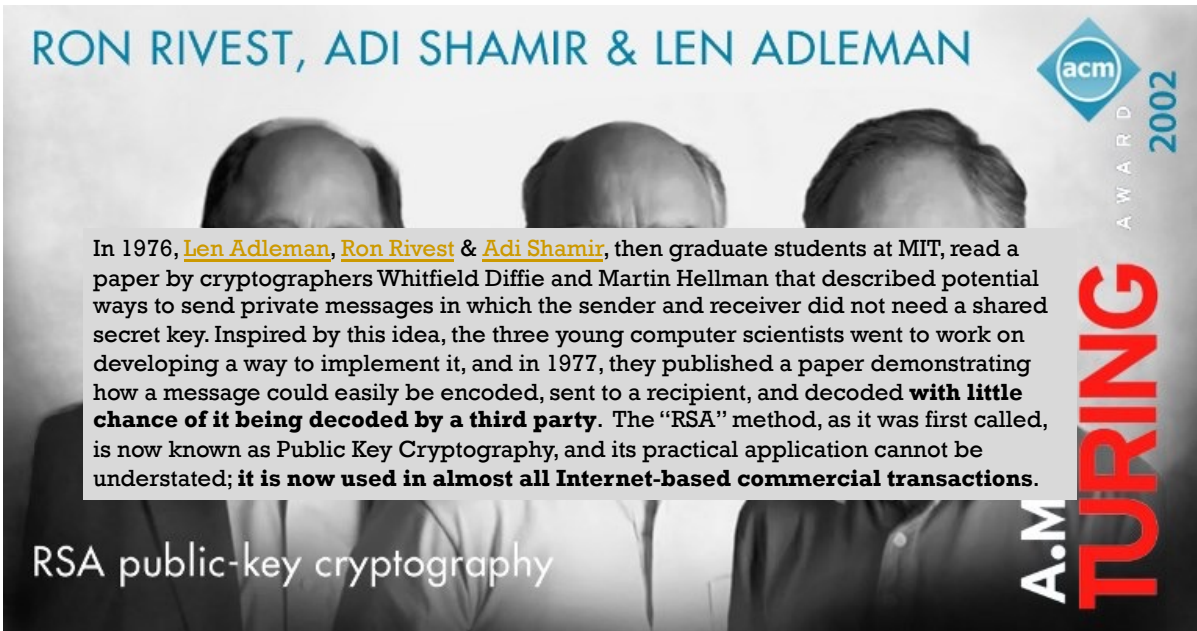
A.M. TURING

RSA public-key cryptography

S. Ranise - Security & Trust (FBK)

60

60



RON RIVEST, ADI SHAMIR & LEN ADLEMAN

acm
AWARD
2002

A.M. TURING

RSA public-key cryptography

In 1976, [Len Adleman](#), [Ron Rivest](#) & [Adi Shamir](#), then graduate students at MIT, read a paper by cryptographers Whitfield Diffie and Martin Hellman that described potential ways to send private messages in which the sender and receiver did not need a shared secret key. Inspired by this idea, the three young computer scientists went to work on developing a way to implement it, and in 1977, they published a paper demonstrating how a message could easily be encoded, sent to a recipient, and decoded **with little chance of it being decoded by a third party**. The “RSA” method, as it was first called, is now known as Public Key Cryptography, and its practical application cannot be understated; **it is now used in almost all Internet-based commercial transactions.**

S. Ranise - Security & Trust (FBK)

61

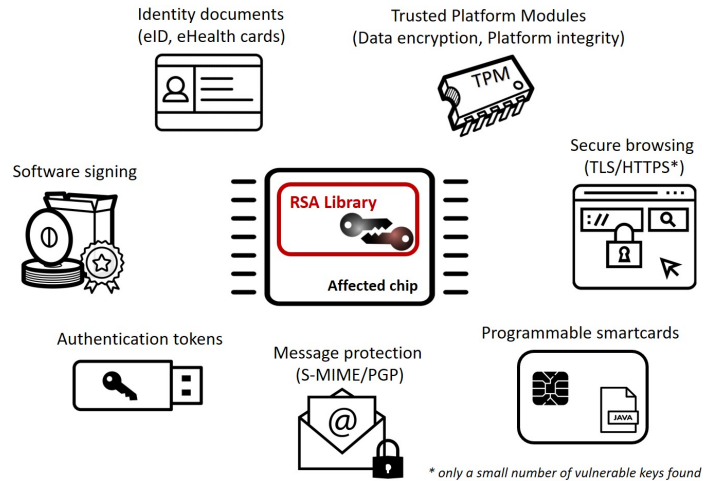
61

A RECENT PROBLEM WITH RSA

M. Nemecek, M. Sys, P. Svenda, D. Klinec, V. Matyas: The Return of Coppersmith's Attack..., ACM CCS 2017

The usage domains affected by the vulnerable library

- **Vulnerability in generation of RSA keys** used by a software library adopted in cryptographic smartcards, security tokens and other secure hardware chips manufactured by Infineon Technologies AG
- It allows for a **practical factorization attack**, in which the attacker computes the private part of an RSA key
- **Attack is feasible for commonly used key lengths, including 1024 and 2048 bits**, and affects chips manufactured as early as 2012, that are now commonplace



62

A BRIEF OVERVIEW OF THE VULNERABILITY

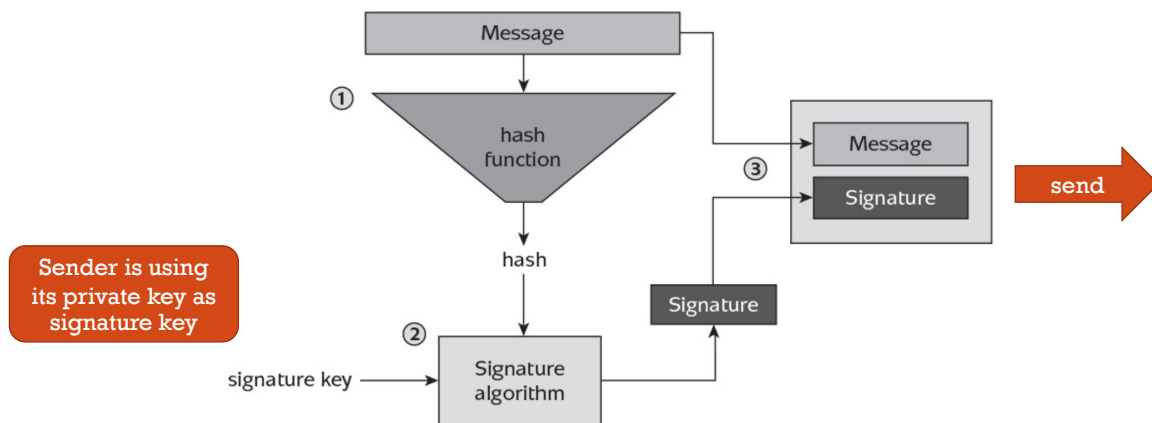
- **Algorithmic vulnerability** characterized by a specific structure of the generated RSA primes, which makes factorization of commonly used key lengths including 1024 and 2048 bits practically possible
- Only the knowledge of a public key is necessary and no physical access to the vulnerable device is required
- The vulnerability does NOT depend on a weak or a faulty random number generator - all RSA keys generated by a vulnerable chip are impacted
- The attack was practically verified for several randomly selected 1024-bit RSA keys and for several selected 2048-bit keys
 - The worst-case price of the factorization on an Amazon AWS c4 computation instance is \$76 for the 1024-bit key and about \$40,000 for the 2048-bit key
- **IMPACT:** Remote attacker can compute an RSA private key from the value of a public key
 - Private key can be misused for impersonation of a legitimate owner, decryption of sensitive messages, forgery of signatures (such as for software releases) and other related attacks
 - Currently confirmed number of vulnerable keys found is about 760,000 but possibly up to two to three magnitudes more are vulnerable

For more details, see https://crocs.fi.muni.cz/public/papers/rsa_ccs17

63

INTEGRITY WITH RSA (1)

The hash function is used mainly because of efficiency reasons since RSA is relatively heavy from a computational point of view

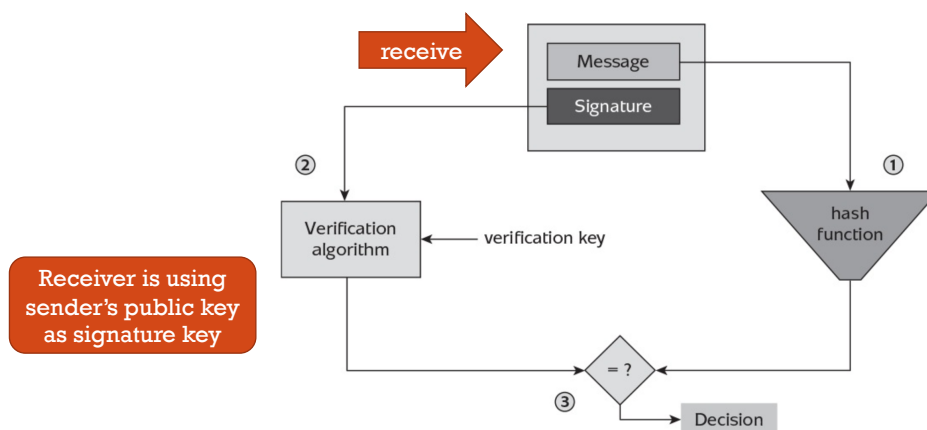


64

64

INTEGRITY WITH RSA (2)

It is important to observe that integrity is guaranteed under the assumption that only a designated entity is in control of a signature key, i.e. the private key of the sender in the case of RSA



65

65

RSA FOR KEY EXCHANGE

Assumptions

- Alice has generated a pair K_{apr} and K_{apu} of private and public key, respectively
- Bob has generated a pair K_{bpr} and K_{bpu} of private and public key, respectively
- **Alice knows that K_{bpu} is the public key of Bob**
- **Bob knows that K_{apu} is the public key of Alice**

Eve may intercept and tamper any message sent over the insecure channel without breaking ciphers

Alice

1. generates a symmetric key K_{ab}
2. $D_{ab} = \text{hash}(K_{ab})$
3. $S_{ab} = \text{enc}(K_{apr}, D_{ab})$ **integrity**
4. $M = \text{enc}(K_{bpu}, (K_{ab}, S_{ab}))$ **confidentiality**
5. sends message M to Bob

Bob

1. receives M'
2. $(K_{ab}', S_{ab}') = \text{dec}(K_{bpr}, M')$
3. $D_{ab}' = \text{hash}(K_{ab}')$
4. $D_{ab}'' = \text{dec}(K_{apu}, S_{ab}')$
5. if $D_{ab}' = D_{ab}''$ then use K_{ab} for data exchange, otherwise discard it



66

66

PKC ALGORITHMS USED TODAY

▪ RSA (Rivest, Shamir, Adleman)

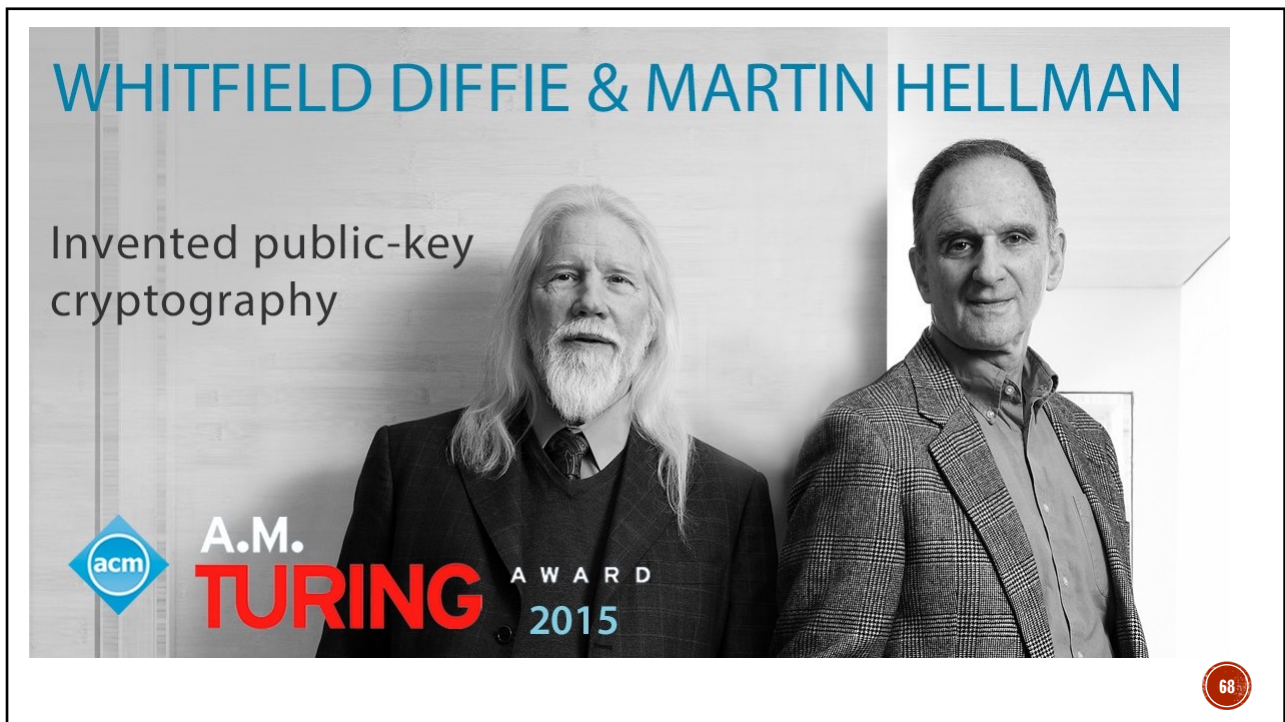
- Used in hundreds of software products and can be used for **key exchange, digital signatures, or encryption** of small blocks of data
- Mathematical "trick" of RSA: relatively easy to compute product compared to computing factorizations
- RSA uses a variable size encryption block and a variable size key
- Key-pair is derived from a very large number, n , that is the product of two prime numbers chosen according to special rules
 - Primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors
 - **An attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure**
 - The ability for computers to factor large numbers, and therefore attack RSA scheme, is rapidly improving and systems today can find the prime factors of numbers with more than 200 digits
 - Nevertheless, if a large number is created from two prime factors that are roughly the same size, there is no known factorization algorithm that will solve the problem in a reasonable amount of time; a 2005 test to factor a 200-digit number took 1.5 years and over 50 years of compute time

▪ DH (Diffie-Hellman)

- After the RSA algorithm was published, Diffie and Hellman came up with their own algorithm
- DH is **used for secret-key key exchange only**, not for authentication or digital signatures

67

67



68



69

DH: AN OVERVIEW

- Mathematical "trick" of DH:
 - Relatively easy to compute exponents compared to computing discrete logarithms
 - See next slide about basic mathematical notions
- DH allows two parties, say Alice and Bob, to generate a secret key
 - **Exchange information over an unsecure communications channel to perform calculation but eavesdropper, Eve, cannot determine shared secret key based upon this information**
- Flow (see next next slide)

70

70

BASIC MATH NOTIONS

- **Finite field $F = GF(p)$** = Integers modulo prime p
 - GF stands for Galois Field
- Exponentiation $x \rightarrow g^x \pmod{p}$ is the one-way function
 - This is *computationally easy* to compute
- **Discrete Logarithm Problem (DLP)**
 - Given p, g, X , find the discrete logarithm x so that $X = g^x \pmod{p}$
 - This is a *computationally difficult* problem to solve for **carefully chosen p, g, X** :
 - p is a prime
 - g is a **generator**, i.e. for $0 < X < p$ there is x s.t. $X = g^x \pmod{p}$
- Important property: exponentiation and modulus commute

71

71

DH: AN OVERVIEW

Alice...



1. Choose a large (random) number a
 - this is Alice **private** key
2. Compute $A = g^a \bmod p$
 - this is Alice **public** key
3. Exchange public key with Bob
4. Compute $K_A = g^{(a*b)} \bmod p$

Bob...



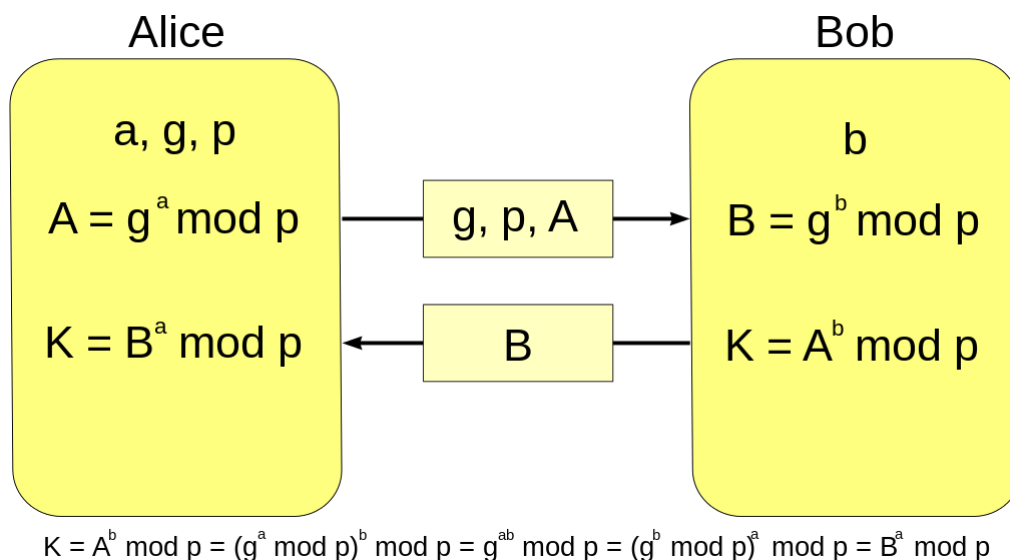
1. Choose a large (random) number b
 - this is Bob **private** key
2. Compute $B = g^b \bmod p$
 - this is Bob **public** key
3. Exchange public key with Alice
4. Compute $K_B = g^{(b*a)} \bmod p$

- **a and b are kept secret while A and B are openly shared**; these are the private and public keys, respectively
- Based on their own private key and the public key learned from the other party, Alice and Bob have computed their secret keys, K_A and K_B , respectively, which are **both** equal to $g^{(a*b)} \bmod N$

72

72

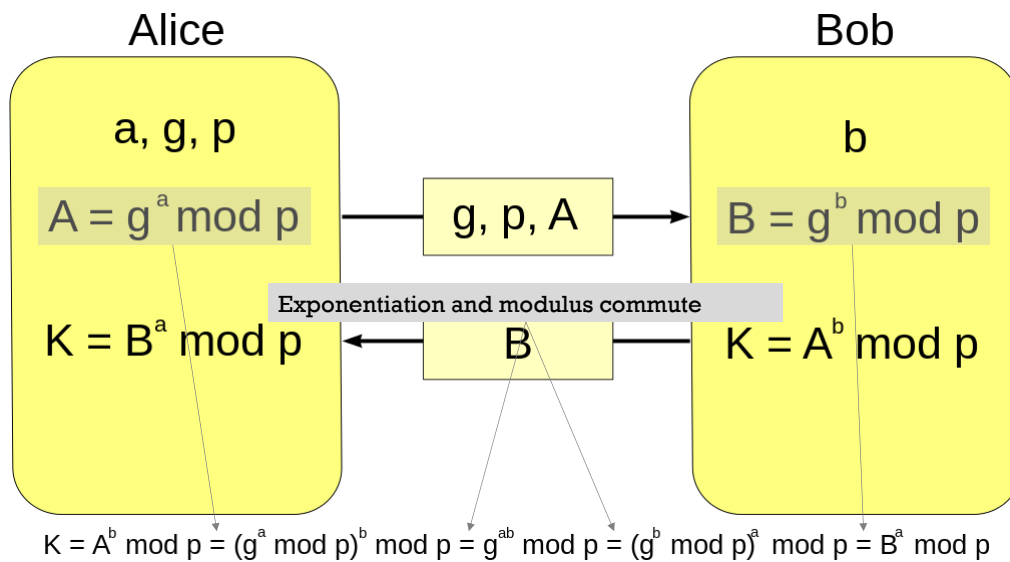
DH EXPLAINED IN A PICTURE



73

73

DH EXPLAINED IN A PICTURE



74

74

DH SECURITY CONSIDERATIONS

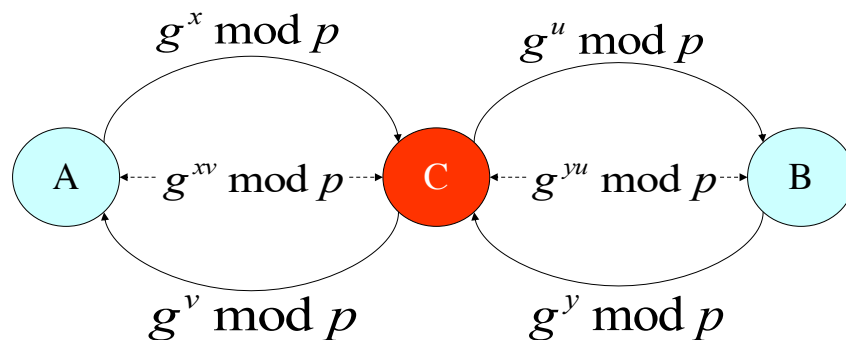
- Security of DH depends on the difficulty of the DLP
 - an attacker able to solve DLP could obtain X_A and X_B from the public keys of A and B, namely $Y_A = G^{X_A} \bmod N$ and $Y_B = G^{X_B} \bmod N$
- The security of the DH protocol is **not known** to be equivalent to the DLP
- DH is a key agreement protocol
 - Secrecy:** assuming that DHP is difficult, an attacker observing the messages exchanged does not learn the key

75

75

DH SECURITY CONSIDERATIONS (CONT'D)

- **DH does not provide authentication**
 - Parties do not know whom they are establishing a key with
- An attacker **C** sitting between **A** and **B** can mount a Man-In-The-Middle (MITM) attack:



76

76

TAKEAWAYS

- Avoid to design your own crypto algorithms
 - **Amateur designs are usually broken quite easily**
- DES should no longer be used
 - An extension of DES (Triple DES using 3 keys and invokes 3 times DES) still used in the financial sector
- Recommended key length for block cypher: at least 80/90 bits
 - Use AES
- No provable security for DES and AES
 - **Only resistant to known attacks** (aiming to reduce the search space of a brute force attack)
- Don't forget key management!
 - *How to share the key?* **Answer:** key distribution techniques (see next slide)
 - *How to keep the key secret?* **Answer:** use access control techniques
 - With n participants, n^2 keys are needed

77

77

TAKEAWAYS-CONT'D

- PKC \neq RSA: some specific RSA properties do not hold for any PKC system
- **Provable security** by **reduction proofs to open problems**: factoring or DLP
- **Concrete security depends on the state of the art in solving those problems**
- DH: only used for key exchange but problems with authenticity
 - Beware of **MITM** attack!
- Anthropomorphic metaphors may mislead: **stories about Alice & Bob create the illusion that you talk about persons rather than keys**

78

78

RECAP QUESTIONS

- What is a cryptosystem?
- Why key management is crucial for cryptography?
- What does it mean for a cryptographic technique to be computationally secure?
- What are substitution and transposition ciphers? Given an example for each one.
- What is symmetric cryptography?
- What are block and stream ciphers?
- Why is DES deprecated? Why is AES still used?

79

79

RECAP QUESTIONS-CONT'D

- What is asymmetric (or public key) cryptography?
- What are the main advantages and disadvantages of symmetric and asymmetric cryptography?
- What is RSA and for what it can be used?
- What is Diffie-Hellman and for what it can be used?
 - What is a possible attack to Diffie-Hellman?
- What is a one-way function?
 - What is the one-way function used in RSA?
 - What is the one-way function used in Diffie-Hellman?

80

80

RELATIVE SECURITY PER KEY SIZE

Method	Date	Symmetric	Factoring Modulus	Discrete Logarithm Key Group		Elliptic Curve	Hash
Lenstra / Verheul	2021	86	1937 1536	153	1937	163	172
Lenstra Updated	2021	82	1416 1614	164	1416	164	164
ECRYPT	2018 - 2028	128	3072	256	3072	256	256
NIST	2019 - 2030	112	2048	224	2048	224	224
ANSSI	2021 - 2030	128	2048	200	2048	256	256
NSA	-	256	3072	-	-	384	384
RFC3766	-	-	-	-	-	-	-
BSI	2020 - 2022	128	2000	250	2000	250	256

All key sizes are provided in bits. These are the minimal sizes for security.

<https://www.keylength.com/>

81

81