LECTURE 2

General introduction into cryptography

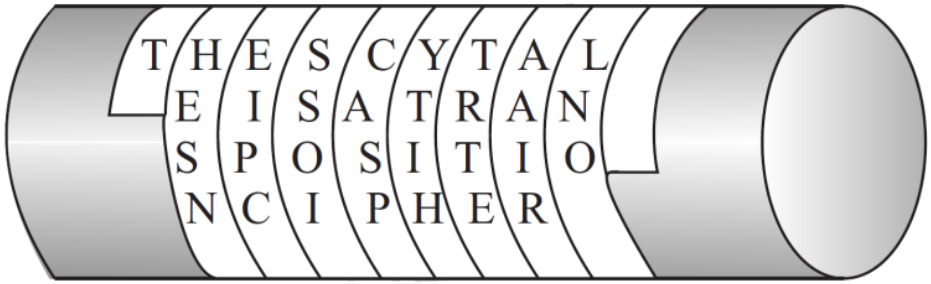
Telecommunication systems department

Lecturer: assistant professor Persikov Anatoliy Valentinovich

OVERVIEW OF CRYPTOLOGY

If we hear the word cryptography our first associations might be e-mail encryption, secure website access, smart cards for banking applications or code breaking during World War II, such as the famous attack against the German Enigma encryption machine.

Cryptography seems closely linked to modern electronic communication. However, cryptography is a rather old business, with early examples dating back to about 2000 B.C., when non-standard “secret” hieroglyphics were used in ancient Egypt. Since Egyptian days cryptography has been used in one form or the other in many, if not most, cultures that developed written language. For instance, there are documented cases of secret writing in ancient Greece, namely the **scytale of Sparta**, or the famous **Caesar cipher** in ancient Rome.



SCYTALE

CRYPTOLOGY BRANCHES

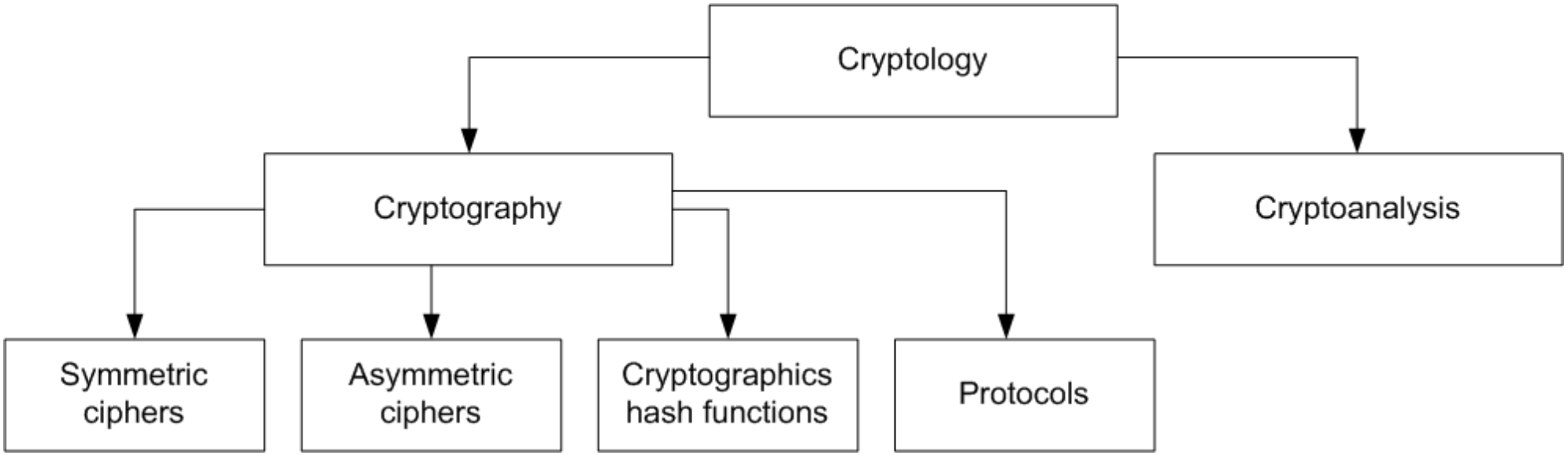
Cryptology splits into two main branches:

1) **Cryptography** is the science of secret writing with the goal of hiding the meaning of a message.
2) **Cryptanalysis** is the science and sometimes art of breaking cryptosystems.

You might think that code breaking is for the intelligence community or perhaps organized crime, and should not be included in a serious classification of a scientific discipline. However, most cryptanalysis is done by respectable researchers in academia nowadays. Cryptanalysis is of central importance for modern cryptosystems: without people who try to break our cryptomethods, we will never know whether they are really secure or not.

Because cryptanalysis is the only way to assure that a cryptosystem is secure, it is an integral part of cryptology.

CRYPTOLOGY BRANCHES



Cryptography itself splits into four main branches:

1) Symmetric algorithms.
2) Asymmetric (or Public-Key) algorithms.
3) Hash functions.
4) Cryptographic protocols.

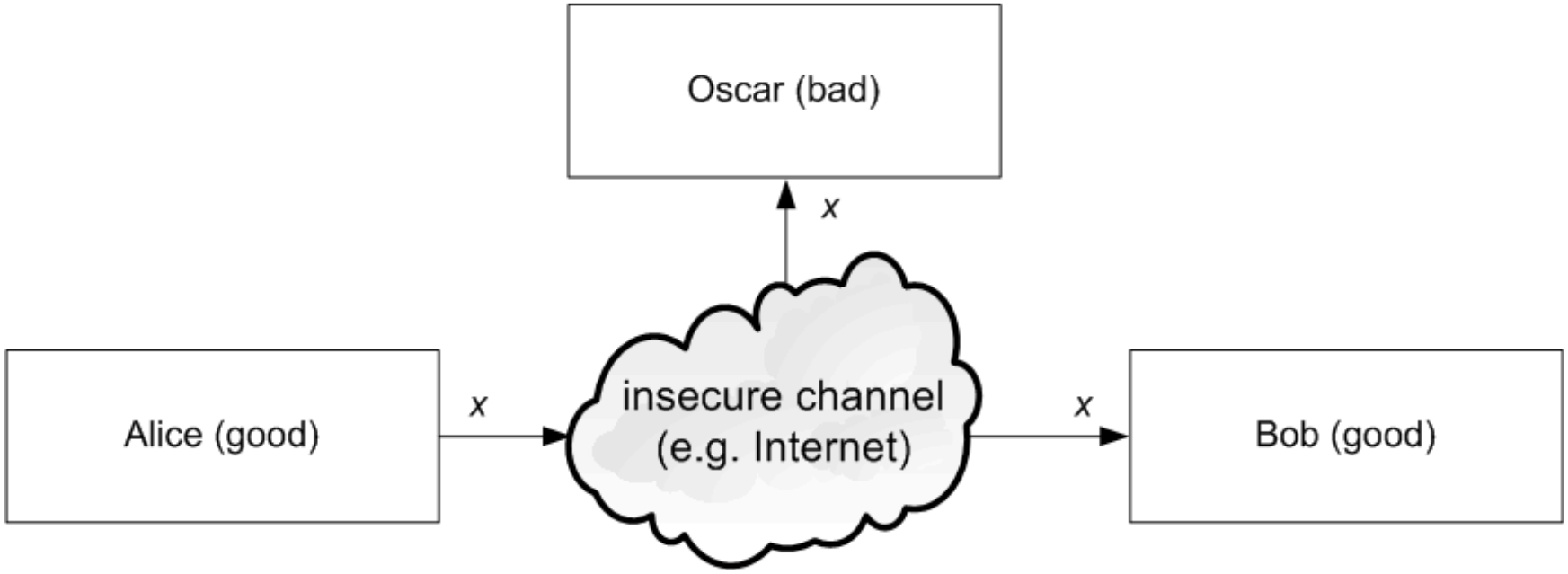
CRYPTOLOGY BRANCHES

1) **Symmetric algorithms** are what many people assume cryptography is about: two parties have an encryption and decryption method for which they share a secret key. All cryptography from ancient times until 1976 was exclusively based on symmetric methods. Symmetric ciphers are still in widespread use, especially for data encryption and integrity check of messages.
2) **Asymmetric (or public-key) algorithms**. In 1976 an entirely different type of cipher was introduced by Whitfield Diffie, Martin Hellman and Ralph Merkle. In public-key cryptography, a user possesses a secret key as in symmetric cryptography but also a public key. Asymmetric algorithms can be used for applications such as digital signatures and key establishment, and also for classical data encryption.
3) **Hash functions** form a third class of algorithms but at the same time they share some properties with symmetric ciphers.
4) **Cryptographic protocols**. Roughly speaking, crypto protocols deal with the application of cryptographic algorithms. Symmetric and asymmetric algorithms can be viewed as building blocks with which applications such as secure Internet communication can be realized. The Transport Layer Security (TLS) scheme, which is used in every Web browser, is an example of a cryptographic protocol.

SYMMETRIC CRYPTOGRAPHY

Symmetric cryptographic schemes are also referred to as symmetric-key, secret-key, and single-key schemes or algorithms.

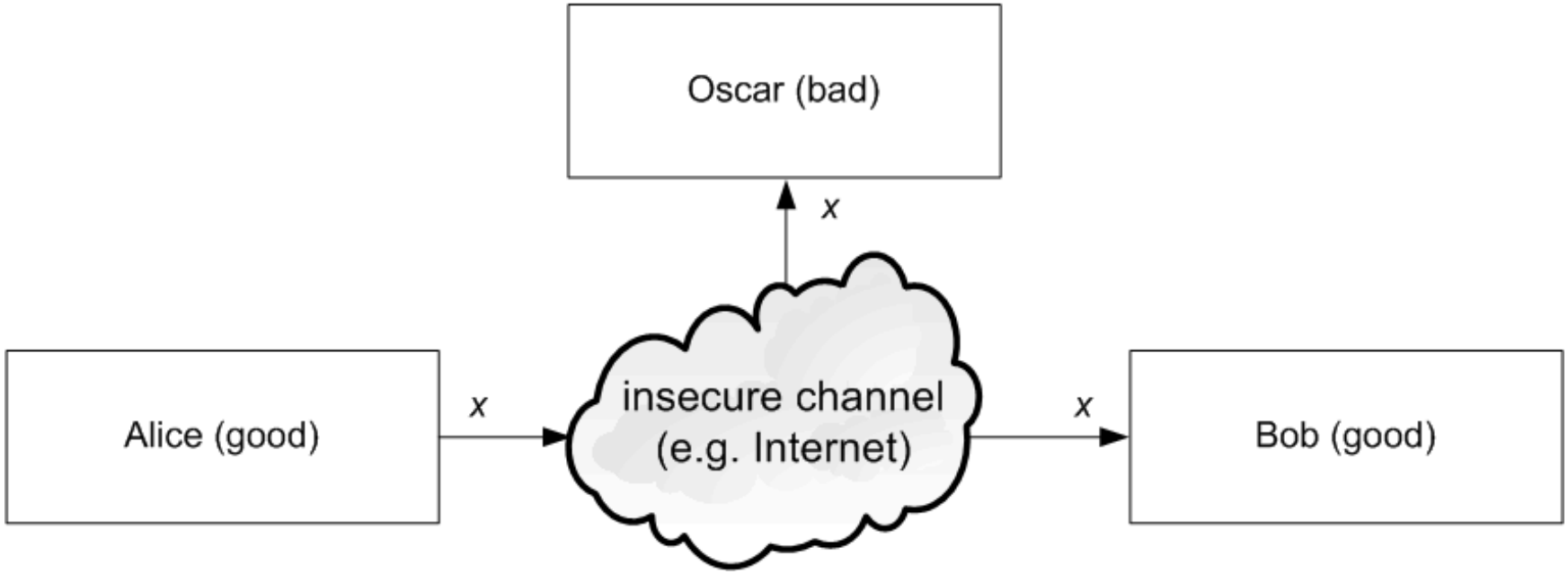
Symmetric cryptography is best introduced with an easy to understand problem: There are two users, Alice and Bob, who want to communicate over an insecure channel.



SYMMETRIC CRYPTOGRAPHY

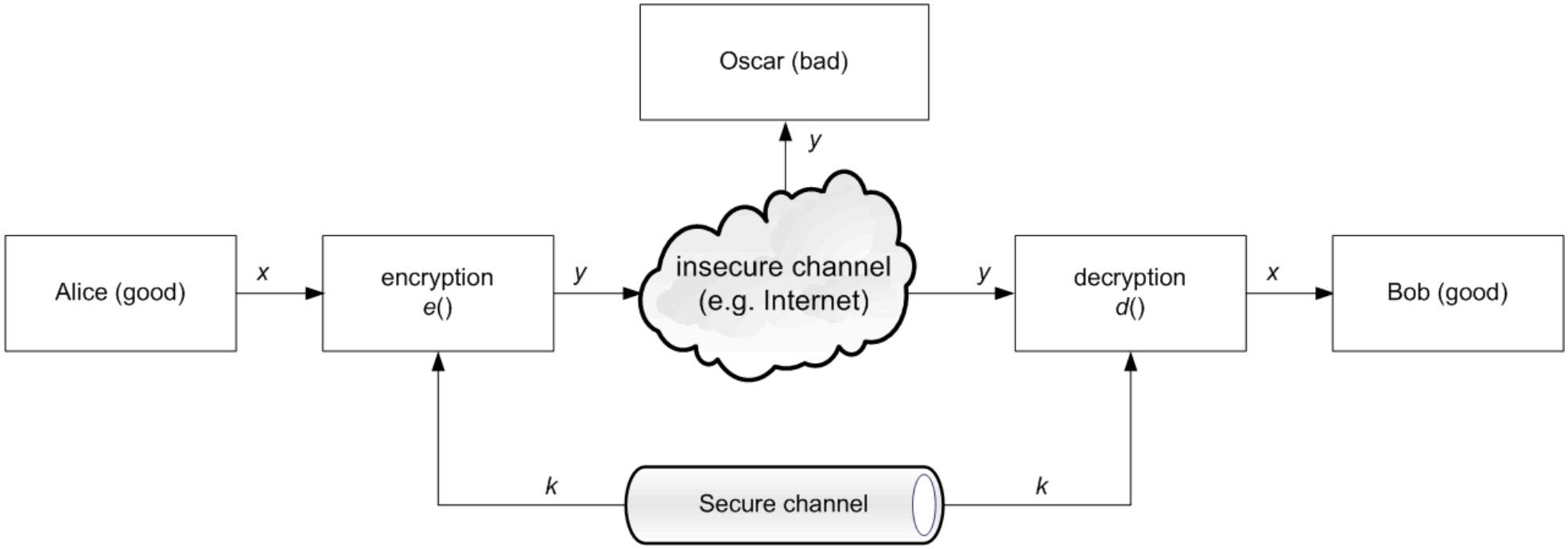
The term channel might sound a bit abstract but it is just a general term for the communication link: This can be the Internet, a stretch of air in the case of mobile phones or wireless LAN communication, or any other communication media you can think of. The actual problem starts with the bad guy, Oscar, who has access to the channel, for instance, by hacking into an Internet router or by listening to the radio signals of a Wi-Fi communication.

This type of unauthorized listening is called **eavesdropping**.



SYMMETRIC CRYPTOGRAPHY

In this situation, symmetric cryptography offers a powerful solution: Alice encrypts her message x using a symmetric algorithm, yielding the ciphertext y. Bob receives the ciphertext and decrypts the message. Decryption is, thus, the inverse process of encryption. What is the advantage? If we have a strong encryption algorithm, the ciphertext will look like random bits to Oscar and will contain no information whatsoever that is useful to him.



SYMMETRIC CRYPTOGRAPHY

The variables x, y and k are important in cryptography and have special names:

1) x is called **plaintext** or **cleartext**,
2) y is called **ciphertext**,
3) k is called **the key**,
4) the set of all possible keys is called **the key space**.

The system needs a secure channel for distribution of the key between Alice and Bob. The secure channel can, for instance, be a human who is transporting the key in a wallet between Alice and Bob. This is, of course, a somewhat cumbersome method. An example where this method works nicely is the pre-shared keys used in Wi-Fi Protected Access (WPA) encryption in wireless LAN.

One important and also counterintuitive fact in this situation is that both the encryption and the decryption algorithms are publicly known. It seems that keeping the encryption algorithm secret should make the whole system harder to break. However, secret algorithms also mean untested algorithms: The only way to find out whether an encryption method is strong, i.e., cannot be broken by a determined attacker, is to make it public and have it analyzed by other cryptographers.

Only thing that should be kept secret in a sound cryptosystem is the key.

SIMPLE SYMMETRIC ENCRYPTION: THE SUBSTITUTION CIPHER

We will now learn one of the simplest methods for encrypting text, the **substitution** (= **replacement**) cipher. Historically this type of cipher has been used many times, and it is a good illustration of basic cryptography. We will use the substitution cipher for learning some important facts about key lengths and about different ways of attacking ciphers.

The goal of the substitution cipher is the encryption of text (as opposed to bits in modern digital systems). The idea is very simple: We substitute each letter of the alphabet with another one.

$A \to k$
$B \to d$
$C \to w$

For instance, the pop group **ABBA** would be encrypted as **kddk**.

We assume that we choose the substitution table completely randomly, so that an attacker is not able to guess it. Note that the substitution table is the key of this cryptosystem. As always in symmetric cryptography, the key has to be distributed between Alice and Bob in a secure fashion.

FIRST ATTACK: BRUTE-FORCE OR EXHAUSTIVE KEY SEARCH

Brute-force attacks are based on a simple concept: Oscar, the attacker, has the ciphertext from eavesdropping on the channel and happens to have a short piece of plaintext, e.g., the header of a file that was encrypted. Oscar now simply decrypts the first piece of ciphertext with all possible keys. Again, the key for this cipher is the substitution table. If the resulting plaintext matches the short piece of plaintext, he knows that he has found the correct key.

Definition. Basic exhaustive key search or brute-force attack.

Let (x, y) denote the pair of plaintext and ciphertext, and let $K = \{k_1, \ldots, k_{\mathcal{K}}\}$ be the key space of all possible keys k_i. A brute-force attack now checks for every $k_i \in K$ if

$$d_{k_i}(y) = x$$

If the equality holds, a possible correct key is found; if not, proceed with the next key.

In practice, a brute-force attack can be more complicated because incorrect keys can give false positive results.

SECOND ATTACK: LETTER FREQUENCY ANALYSIS

First we note that the brute-force attack from above treats the cipher as a black box, i.e., we do not analyze the internal structure of the cipher. The substitution cipher can easily be broken by such an analytical attack.

The major weakness of the cipher is that each plaintext symbol always maps to the same ciphertext symbol. That means that the statistical properties of the plaintext are preserved in the ciphertext. If we go back to the second example we observe that the letter q occurs most frequently in the text. From this we know that q must be the substitution for one of the frequent letters in the English language. For practical attacks, the following properties of language can be exploited:

1) Determine the frequency of every ciphertext letter. The frequency distribution, often even of relatively short pieces of encrypted text, will be close to that of the given language in general. In particular, the most frequent letters can often easily be spotted in ciphertexts. For instance, in English E is the most frequent letter (about 13%), T is the second most frequent letter (about 9%), A is the third most frequent letter (about 8%), and so on. Table lists the letter frequency distribution of English.

SECOND ATTACK: LETTER FREQUENCY ANALYSIS

Letter	Frequency	Letter	Frequency
A	0.0817	N	0.0675
B	0.0150	O	0.0751
C	0.0278	P	0.0193
D	0.0425	Q	0.0010
E	0.1270	R	0.0599
F	0.0223	S	0.0633
G	0.0202	T	0.0906
H	0.0609	U	0.0276
I	0.0697	V	0.0098
J	0.0015	W	0.0236
K	0.0077	X	0.0015
L	0.0403	Y	0.0197
M	0.0241	Z	0.0007

STATISTICS IS STABLE FOR MILLIONS SYMBOLS

SECOND ATTACK: LETTER FREQUENCY ANALYSIS

Let's look at another ciphertext:

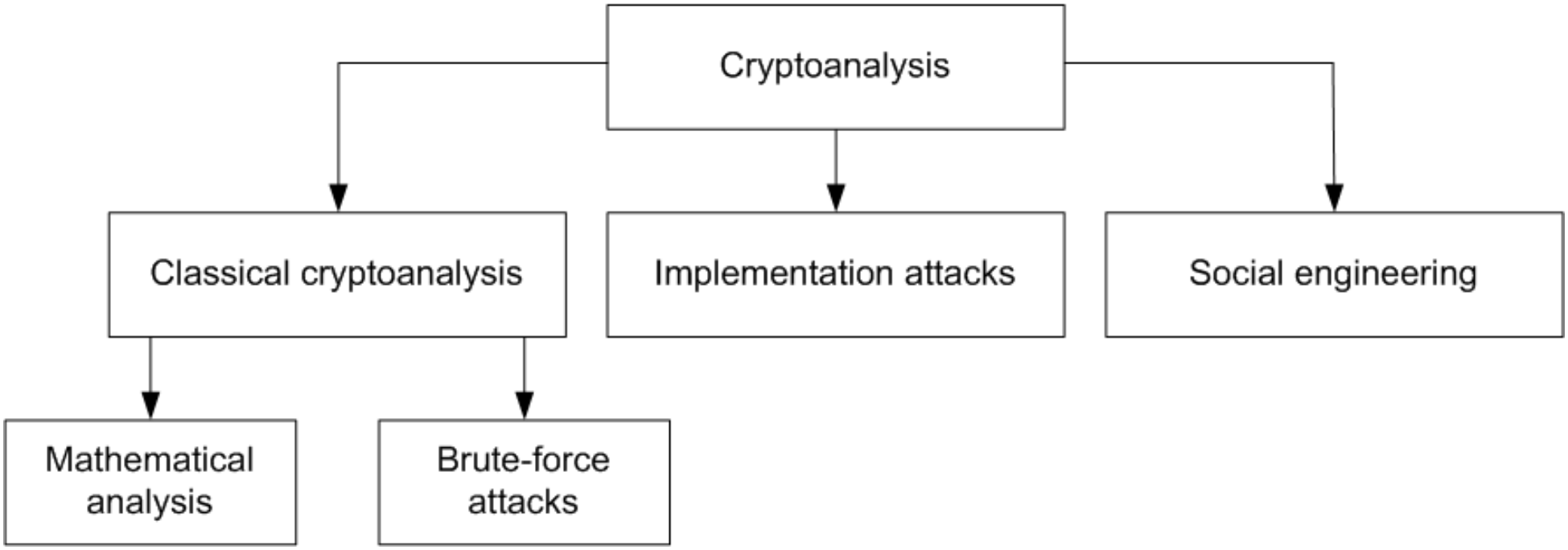
iq ifcc vqqr fb rdq vfllcq na rdq cfjwhwz hr bnnb
hcc hwwhbsqvqbre hwq vhlq

If we analyze the encrypted text from example, we obtain:

WE WILL MEET IN THE MIDDLE OF THE LIBRARY AT NOON
ALL ARRANGEMENTS ARE MADE

Good ciphers should hide the statistical properties of the encrypted plaintext. The ciphertext symbols should appear to be random. Also, a large key space alone is not sufficient for a strong encryption function.

GENERAL THOUGHTS ON BREAKING CRYPTOSYSTEMS



Classical cryptanalysis. Classical cryptanalysis is understood as the science of recovering the plaintext x from the ciphertext y, or, alternatively, recovering the key k from the ciphertext y. We recall from the earlier discussion that cryptanalysis can be divided into analytical attacks, which exploit the internal structure of the encryption method, and brute-force attacks, which treat the encryption algorithm as a black box and test all possible keys.

GENERAL THOUGHTS ON BREAKING CRYPTOSYSTEMS

Implementation Attacks. Side-channel analysis can be used to obtain a secret key, for instance, by measuring the electrical power consumption of a processor which operates on the secret key. The power trace can then be used to recover the key by applying signal processing techniques. In addition to power consumption, electromagnetic radiation or the runtime behavior of algorithms can give information about the secret key and are, thus, useful side channels. Note also that implementation attacks are mostly relevant against cryptosystems to which an attacker has physical access, such as smart cards. In most Internet-based attacks against remote systems, implementation attacks are usually not a concern.

Social Engineering Attacks. Bribing, blackmailing, tricking or classical espionage can be used to obtain a secret key by involving humans. For instance, forcing someone to reveal his/her secret key, e.g., by holding a gun to his/her head can be quite successful. Another, less violent, attack is to call people whom we want to attack on the phone, and say: “This is the IT department of your company. For important software updates we need your password”. It is always surprising how many people are naive enough to actually give out their passwords in such situations.

GENERAL THOUGHTS ON BREAKING CRYPTOSYSTEMS

This list of attacks against cryptographic system is certainly not exhaustive. For instance, buffer overflow attacks or malware can also reveal secret keys in software systems. You might think that many of these attacks, especially social engineering and implementation attacks, are “unfair,” but there is little fairness in real-world cryptography. If people want to break your IT system, they are already breaking the rules and are, thus, unfair. The major point to learn here is:

An attacker always looks for the weakest link in your cryptosystem. That means we have to choose strong algorithms and we have to make sure that social engineering and implementation attacks are not practical.

Even though both implementation attacks and social engineering attacks can be quite powerful in practice, this book mainly assumes attacks based on mathematical cryptanalysis.

Solid cryptosystems should adhere to Kerckhoffs’ Principle, postulated by Auguste Kerckhoffs in 1883:

A cryptosystem should be secure even if the attacker knows all details about the system, with the exception of the secret key. In particular, the system should be secure when the attacker knows the encryption and decryption algorithms.

HOW MANY KEY BITS ARE ENOUGH?

During the 1990s there was much public discussion about the key length of ciphers. Before we provide some guidelines, there are two crucial aspects to remember:

1) The discussion of key lengths for symmetric cryptoalgorithms is only relevant if a brute-force attack is the best known attack. During the security analysis of the substitution cipher, if there is an analytical attack that works, a large key space does not help at all. Of course, if there is the possibility of social engineering or implementation attacks, a long key also does not help.
2) The key lengths for symmetric and asymmetric algorithms are dramatically different. For instance, an 80-bit symmetric key provides roughly the same security as a 1024-bit RSA key.

Table gives a rough indication of the security of symmetric ciphers with respect to brute-force attacks.

Key length	Security estimation
56–64 bits	short term: a few hours or days
112-128 bits	long term: several decades in the absence of quantum computers
256 bits	long term: several decades, even with quantum computers that run the currently known quantum computing algorithms

MOORE LAW

Foretelling the future of course, predicting the future tends to be tricky: We can't really foresee new technical or theoretical developments with certainty. As you can imagine, it is very hard to know what kinds of computers will be available in the year 2030.

For medium-term predictions, **Moore's Law** (Gordon Earle Moore is an American co-founder and Chairman Emeritus of Intel Corporation) is often assumed.

Roughly speaking, Moore's Law states that **computing power doubles every 18 months while the costs stay constant**.

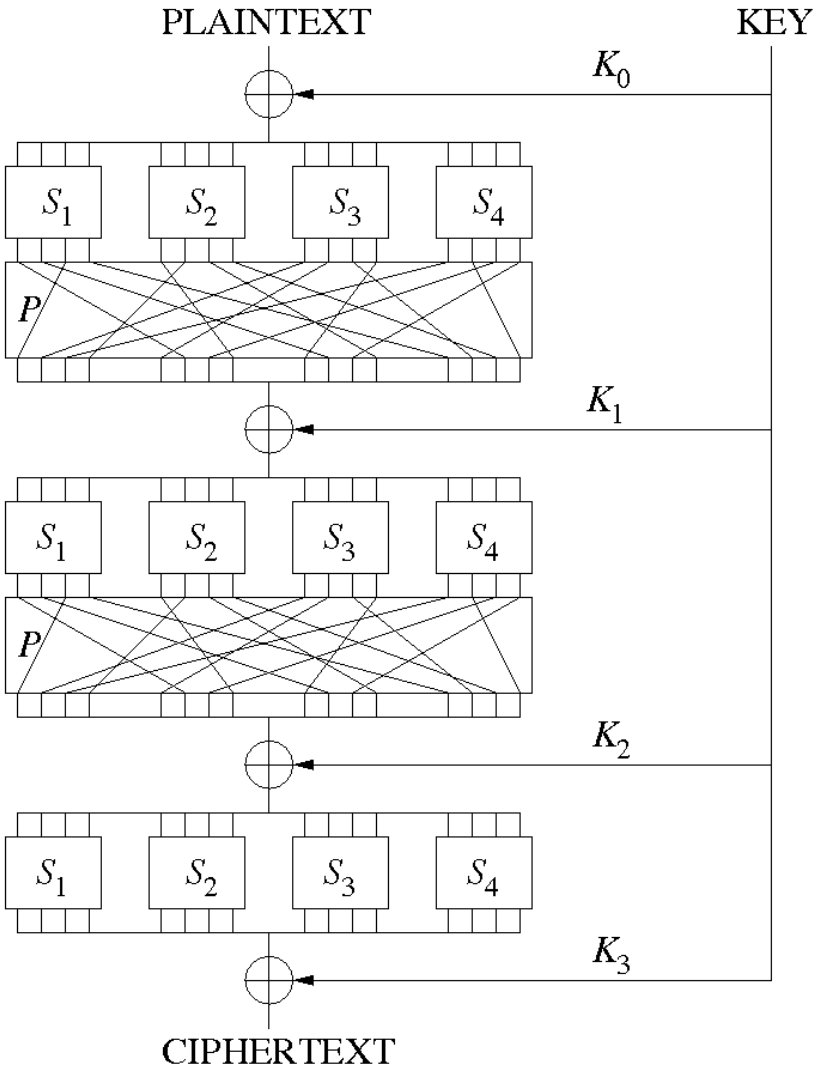
This has the following implications in cryptography: If today we need one month and computers worth $1,000,000 to break a cipher X, then:

- the cost for breaking the cipher will be $500,000 in 18 months (since we only have to buy half as many computers);
- $250,000 in 3 years;
- $125,000 in 4.5 years, and so on.

MOORE LAW

It is important to stress that Moore's Law is an exponential function. In 15 years, i.e., after 10 iterations of computer power doubling, we can do $2^{10} = 1024$ as many computations for the same money we would need to spend today. Stated differently, we only need to spend about 1/1000th of today's money to do the same computation. In the example above that means that we can break cipher X in 15 years within one month at a cost of about \$1,000,000/1024≈\$1000. Alternatively, with \$1,000,000, an attack can be accomplished within 45 minutes in 15 years from now. Moore's Law behaves similarly to a bank account with a 50% interest rate: The compound interest grows very, very quickly. Unfortunately, there are few trustworthy banks which offer such an interest rate.

SUBSTITUTION-PERMUTATION NETWORK



In cryptography, an SP-network, or substitution-permutation network (SPN), is a series of linked mathematical operations used in block cipher algorithms such as AES (Rijndael). Such a network takes a block of the plaintext and the key as inputs, and applies several alternating "**rounds**" or "**layers**" of **substitution boxes** (S-boxes) and **permutation boxes** (P-boxes) to produce the **ciphertext block**. The S-boxes and P-boxes transform (sub-)blocks of input bits into output bits. It is common for these transformations to be operations that are efficient to perform in hardware, such as exclusive or (XOR) and bitwise rotation. The **key** is introduced in each round, usually in the form of "**round keys**" derived from it. (In some designs, the S-boxes themselves depend on the key.)

Decryption is done by simply reversing the process (using the inverses of the S-boxes and P-boxes and applying the round keys in reversed order).

SUBSTITUTION-PERMUTATION NETWORK

An **S-box** substitutes a small block of bits (the input of the S-box) by another block of bits (the output of the S-box):

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

An S-box is usually not at all just a permutation of the bits. Rather, a good S-box will have the property that changing one input bit will change about half of the output bits (or an avalanche effect). It will also have the property that each output bit will depend on every input bit.

SUBSTITUTION-PERMUTATION NETWORK

A **P-box** is a permutation of all the bits: it takes the outputs of all the S-boxes of one round, permutes the bits, and feeds them into the S-boxes of the next round. A good P-box has the property that the output bits of any S-box are distributed to as many S-box inputs as possible.

At each round, the round key (obtained from the key with some simple operations, for instance, using S-boxes and P-boxes) is combined using some group operation, typically XOR.

A single typical S-box or a single P-box alone does not have much cryptographic strength: an S-box could be thought of as a substitution cipher, while a P-box could be thought of as a transposition cipher. However, a well-designed SP network with several alternating rounds of S- and P-boxes already satisfies Shannon's confusion and diffusion properties.

CONFUSION AND DIFFUSION

Avalanche effect refers to a desirable property of cryptographic algorithms, typically block ciphers and cryptographic hash functions. The avalanche effect is evident if, when an input is changed slightly (for example, flipping a single bit) the output changes significantly (e.g., half the output bits flip).

The strict avalanche criterion (SAC) is a generalization of the avalanche effect. It is satisfied if, whenever a single input bit is complemented, each of the output bits changes with a 50% probability.

The reason for **diffusion** is the following: If one changes one bit of the plaintext, then it is fed into an S-box, whose output will change at several bits, then all these changes are distributed by the P-box among several S-boxes, hence the outputs of all of these S-boxes are again changed at several bits, and so on. Doing several rounds, each bit changes several times back and forth, therefore, by the end, the ciphertext has changed completely, in a pseudorandom manner. In particular, for a randomly chosen input block, if one flips the i - th bit, then the probability that the j - th output bit will change is approximately a half, for any i and j, which is the **Strict Avalanche Criterion**.

The reason for **confusion** is exactly the same as for diffusion: changing one bit of the key changes several of the round keys, and every change in every round key diffuses over all the bits, changing the ciphertext in a very complex manner. Vice versa, changing one bit in the ciphertext will change the key completely.

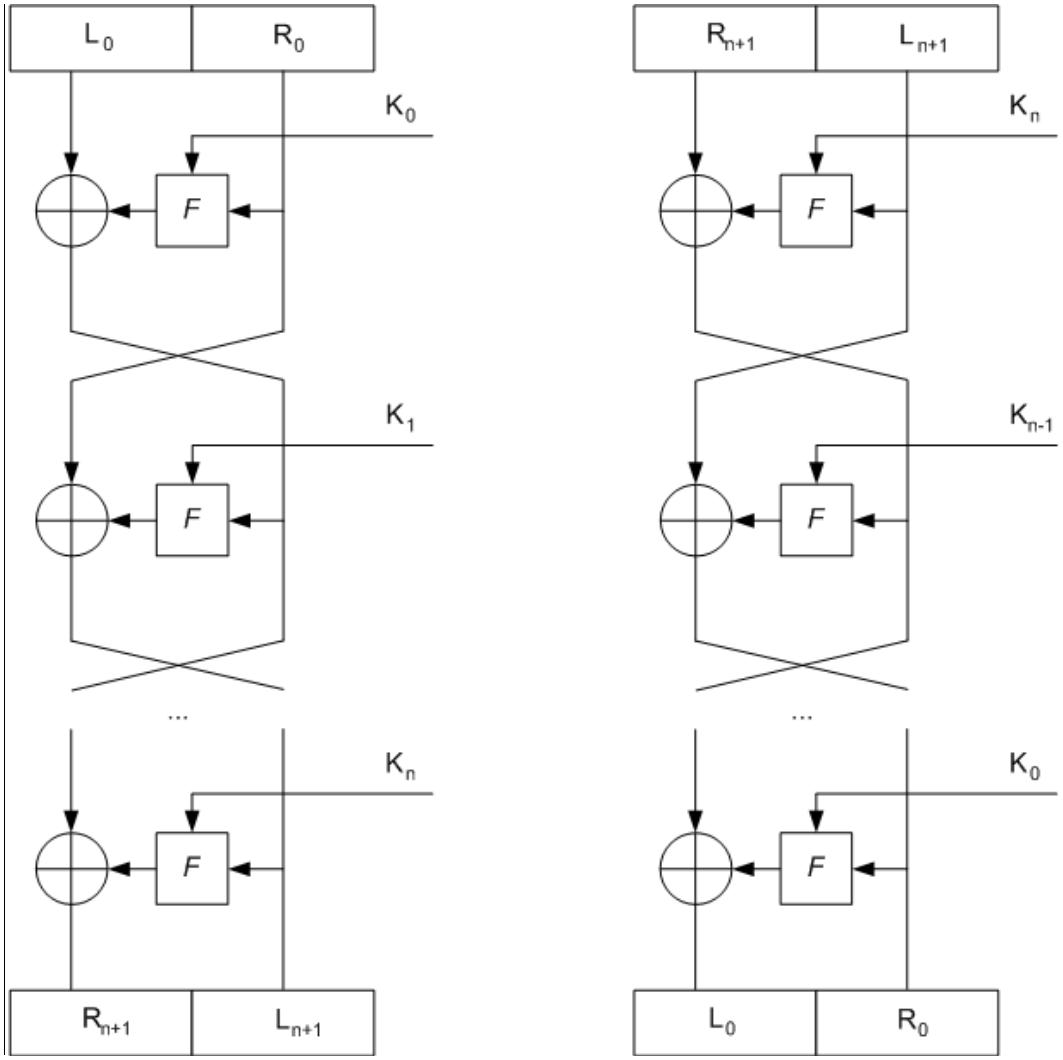
FEISTEL NETWORK

In cryptography, a **Feistel cipher** is a symmetric structure used in the construction of block ciphers, named after the German-born physicist and cryptographer **Horst Feistel** who did pioneering research while working for IBM (USA); it is also commonly known as a **Feistel network**. A large proportion of block ciphers use the scheme, including the Data Encryption Standard (DES). The Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. Therefore the size of the code or circuitry required to implement such a cipher is nearly halved.

A **Feistel network** is an iterated cipher with an internal function called a round function.

Feistel networks were first seen commercially in IBM's Lucifer cipher, designed by Horst Feistel and Don Coppersmith in 1973. Feistel networks gained respectability when the U.S. Federal Government adopted the DES (a cipher based on Lucifer, with changes made by the NSA). Like other components of the DES, the iterative nature of the Feistel construction makes implementing the cryptosystem in hardware easier (particularly on the hardware available at the time of DES' design).

FEISTEL NETWORK



ENCRYPTION **DECRYPTION**

FEISTEL NETWORK MATHEMATICAL DESCRIPTION

Let F be the round function and let K_0, K_1,…,K_n be the sub-keys for the rounds $0,1,\ldots,n$ respectively.

Then the basic operation is as follows:

Split the plaintext block into two equal pieces, (L_0, R_0)

For each round $i = 0,1,\ldots,n$, compute

$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus F(R_i, K_i)$.

Then the ciphertext is (R_{n+1}, L_{n+1}).

Decryption of a ciphertext (R_{n+1}, L_{n+1}) is accomplished by computing for

$i = n, n-1, \ldots, 0$

$R_i = L_{i+1}$

$L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$.

Then (L_0, R_0) is the plaintext again.

One advantage of the Feistel model compared to a substitution-permutation network is that the round function F does not have to be invertible.

THANKS FOR ATTENTION