Introduction to Cryptography

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

What cryptography is about

Cryptography, cryptanalysis, cryptology

- Cryptography = derived from the Greek words kryptós (hidden, secret) and gráphein (to write)

  It is the practice and study of techniques for secure communication in the presence of a third party called adversary

- Cyptanalysis = derived from the Greek words kryptós (hidden,secret) and analýein (to loosen, to untie)

  It is the study of analyzing the hidden aspects of the systems (used to breach cryptographic security systems)

- Cryptology = derived from the Greek words kryptós (hidden, secret) and logia (study)

  It is the scientific study of cryptography and cryptanalysis

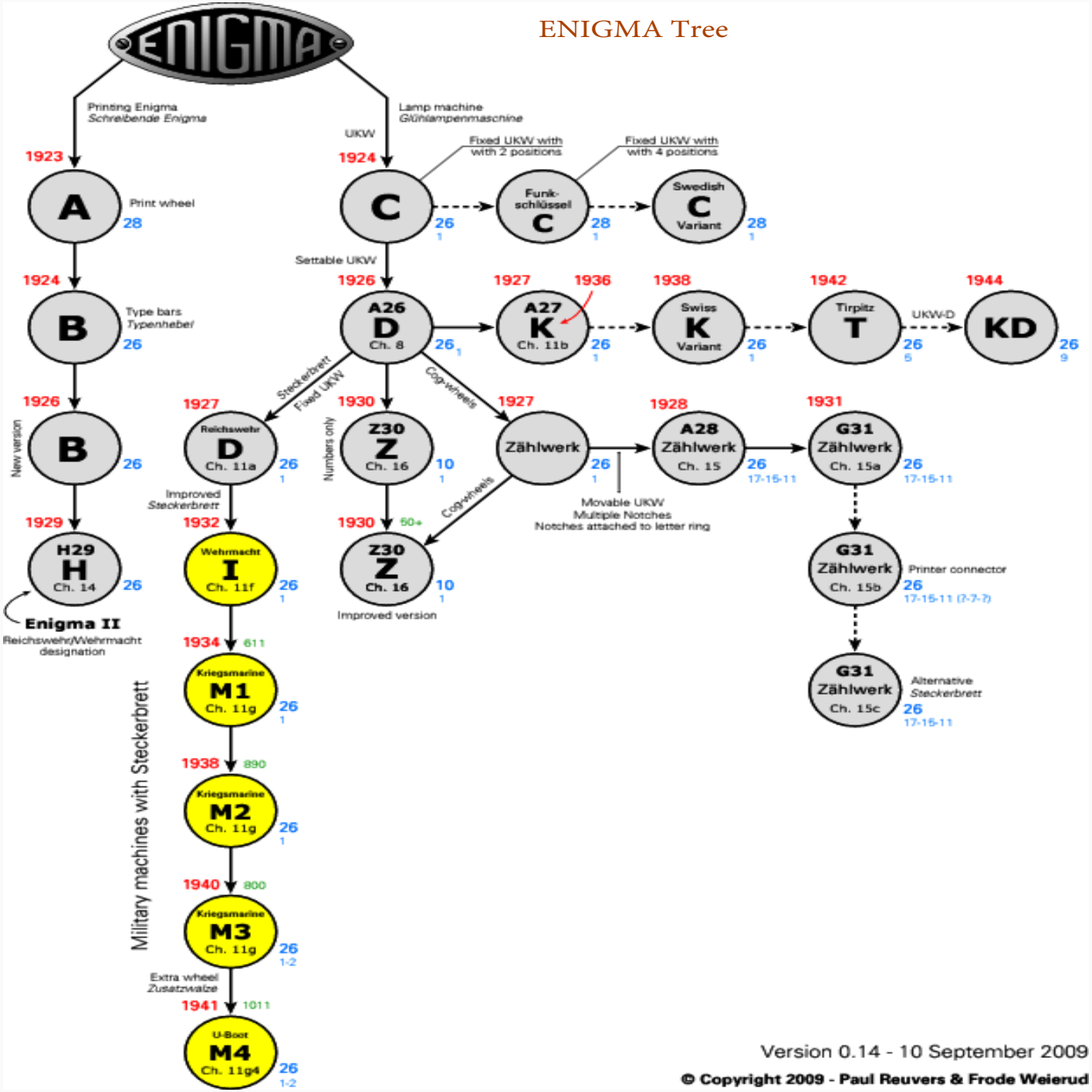
Brief history of cryptography

Classical cryptography

- The oldest forms of cryptography date back to at least Ancient Egypt, when non-standard hieroglyphs were used to communicate (non-standard hieroglyphs were found carved into the wall of a tomb in Egypt from circa 1900 BCE)
- Julius Caesar (100-44 BC) used a simple substitution cipher in his private correspondence, by circularly shifting the letters three positions to the right. This is called nowadays the Caesar cipher
- Thomas Jefferson, the father of American cryptography, invented a cipher wheel in 1795. It was re-invented a century later and used by the US Army from 1923 until 1942 as the M-94 cipher wheel

World war I-II cryptography

- Rotor-based cipher machine = invented by two Dutch naval officers T.A. van Hengel and R.P.C. Sprengler, and produced first in 1915 for the Dutch War Department
- German's Enigma machine = invented first by Arthur Scherbius in 1918, right at the end of World War I. It was produced in many models and variants
- Enigma was broken by a Polish team of mathematicians (M. Rejwski, J. Rózycki and H. Zygalski) and a British team of code-breakers and mathematicians (including Alan Turing)
- Japanese Purple Machine, developed in 1937 using techniques first discovered by Herbert O. Yardley
- W.F. Friedman, the father of American cryptanalysis, led a team which broke the Japanese Purple Code in 1940


ENIGMA Tree
ENIGMA
Printing Enigma
Schreibende Enigma
Lamp machine
Glühlampenmaschine
UKW
1923
A
Print wheel
28
1924
B
Type bars
Typenhebel
26
1926
B
New version
26
1929
H29
H
Ch. 14
26
Enigma II
Reichswehr/Wehrmacht designation
1924
C
26
1
Fixed UKW with with 2 positions
Fixed UKW with with 4 positions
Funk-schlüssel C
28
1
Swedish C Variant
28
1
Settable UKW
1926
A26
D
Ch. 8
26 1
1927
1936
A27
K
Ch. 11b
26
1
1938
Swiss K Variant
26
1
1942
Tirpitz
T
26
5
UKW-D
1944
KD
26
9
Steckerbrett
Fixed UKW
Cog-wheels
1927
Reichswehr
D
Ch. 11a
26
1
Improved Steckerbrett
1932
Wehrmacht
I
Ch. 11f
26
1
1934
611
Kriegsmarine
M1
Ch. 11g
26
1
1938
890
Kriegsmarine
M2
Ch. 11g
26
1
1940
800
Kriegsmarine
M3
Ch. 11g
26
1-2
Extra wheel
Zusatzwalze
1941
1011
U-Boot
M4
Ch. 11g4
26
1-2
Military machines with Steckerbrett
1930
Z30
Z
Ch. 16
10
1
Numbers only
1930
50+
Z30
Z
Ch. 16
10
1
Improved version
Cog-wheels
1927
Zählwerk
26
1
Movable UKW
Multiple Notches
Notches attached to letter ring
1928
A28
Zählwerk
Ch. 15
26
17-15-11
1931
G31
Zählwerk
Ch. 15a
26
17-15-11
G31
Zählwerk
Ch. 15b
Printer connector
26
17-15-11 (2-7-7)
G31
Zählwerk
Ch. 15c
Alternative Steckerbrett
26
17-15-11
Version 0.14 - 10 September 2009
© Copyright 2009 - Paul Reuvers & Frode Weierud

Modern cryptography

- Modern cryptography begins by Claude Shannon's paper Communication Theory of Secrecy Systems in 1949
- In the 1970s, Horst Feistel developed a "family" of ciphers, the Feistel ciphers, used in 1976 to establish FIPS PUB-46 (DES)
- In 1976, Martin Hellman, Whitfield Diffie, and Ralph Merkle, have introduced the concept of public-key cryptography
- In 1977, Ronald L. Rivest, Adi Shamir and Leonard M. Adleman proposed the first public-key cipher, RSA (still secure and in use)
- The Electronic Frontier Foundation (EFF) built the first unclassified hardware for cracking DES (the EFF DES Cracker)
- Oct 2001 – Nov 2002: Advanced Encryption Standard
- Nov 2002 – : lots of modern topics developed around cloud/quantum computing era

Classical ciphers

Cipher

In his seminal paper [10], Shannon introduced the concept of a secrecy system (cipher) as follows:

1. A secrecy system (cipher) is a "family of uniquely reversible transformations ..."
2. Each transformation is associated to some key;
3. The set of keys is finite and a probability distribution on this set is given;
4. There is a finite set of messages on which the transformations can be applied, and a probability distribution on this set is given.

Shannon's definition of a secrecy system (cipher) is suitable enough for illustrative purposes and for the concept of perfect security.

Cipher

Definition 1

A (Shannon) cipher is a 5-tuple $\mathcal{S} = (\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{E}, \mathcal{D})$, where:

1. $\mathcal{K}$ is a non-empty finite set of keys. It is assumed that a probability distribution on $\mathcal{K}$ is also given;
2. $\mathcal{M}$ is a non-empty finite set of messages or plaintexts. It is assumed that a probability distribution on $\mathcal{M}$ is also given;
3. $\mathcal{C}$ is a non-empty finite set of ciphertexts or cryptotexts;
4. $\mathcal{E}$ and $\mathcal{D}$ are two sets of functions (transformations)

   $$\mathcal{E} = \{e_K : \mathcal{M} \to \mathcal{C} | K \in \mathcal{K}\} \quad \text{and} \quad \mathcal{D} = \{d_K : \mathcal{C} \to \mathcal{M} | K \in \mathcal{K}\},$$

   such that $d_K(e_K(x)) = x$, for any $K \in \mathcal{K}$ and $x \in \mathcal{M}$.

e_K is the encryption rule, and d_K is the decryption rule, induced by K.

Cipher

Remark 2

1. *$\mathcal{E}$ and $\mathcal{D}$ in Definition 1 can be viewed as functions:*

$$\mathcal{E}(K, m) = e_K(m), \quad \mathcal{D}(K, c) = d_K(c)$$

This is the way many authors do; however, Shannon emphasized on presenting these transformations as families of functions and we preferred to keep this presentation form. However, we will sometimes adopt the notation

$$\mathcal{E}_K(m) = \mathcal{E}(K, m) \text{ and } \mathcal{D}_K(c) = \mathcal{D}(K, c)$$

2. *Definition 1 does not say anything about the efficiency of computing e_K and d_K. In Shannon's definition of a cipher, these are arbitrary functions (defined on finite sets). However, in practice, these must be efficiently computed.*

Substitution ciphers

A substitution cipher substitute message units (letters, groups of letters etc.) by ciphertext units. Subclasses:

- Monographic (polygraphic): operates on single (groups of) letters;
- Simple: each letter is replaced by another letter (symbol);
- Mono-alphabetic: a single substitution is used to map each message letter (symbol) to a ciphertext letter (symbol);
- Poly-alphabetic: multiple substitutions map message letters (symbols) to ciphertext letters (symbols).

Permutation ciphers: simple substitution cipher where the substitution is a permutation on the message alphabet.

Transposition ciphers

A transposition cipher re-order message units (letters, groups of letters etc.) without changing them.

Example 3

Assume the message is "we are at school now, learning cryptography". We organize the message into a matrix

w	*e*	*a*	*r*	*e*	*a*
t	*s*	*c*	*h*	*o*	*o*
l	*n*	*o*	*w*	*l*	*e*
a	*r*	*n*	*i*	*n*	*g*
c	*r*	*y*	*p*	*t*	*o*
g	*r*	*a*	*p*	*h*	*y*

If we read the columns in the order "6, 1, 5, 2, 4, 3" (which might be the key), we get the following ciphertext:

aoegoywtlacgeolnthesnrrrrhwippaconya

Affine cipher

An affine cipher is a simple substitution cipher where the substitution is an affine function.

Cipher 1 (Affine)

- $\mathcal{M} = \mathcal{C} = V^+$, *where* $V = \{v_0, \ldots, v_{n-1}\}$ *is some given ordered alphabet with* n *letters;*
- $\mathcal{K} = \{(a, b) \in \mathbb{Z}_n \times \mathbb{Z}_n \mid (a, n) = 1\}$;
- *For any key* $K = (a, b)$ *and* $m = v_{i_1} \cdots v_{i_t} \in \mathcal{M}$,

$$e_K(v_{i_1} \cdots v_{i_t}) = v_{j_1} \cdots v_{j_t}$$

where $j_1 = (a \cdot i_1 + b) \bmod n$, *and so on.*
For decryption,

$$d_K(v_{i_1} \cdots v_{i_t}) = v_{j_1} \cdots v_{j_t}$$

where $j_1 = a^{-1}(i_1 - b) \bmod n$, *and so on.*

Affine cipher

Example 4

Assume V is the English small letters alphabet and the letters are numbered from 0 (a) to 25 (z). Therefore, $|V| = 26$.

Let $K = (7, 3)$ and the message "*hot*". Then,

message	h	o	t
index	7	14	19
new index by encryption	0	23	6
ciphertext	a	x	g

Special cases:

1. Shift cipher ($a = 1$).
2. Caesar cipher ($a = 1$ and $b = 3$).

Affine cipher – cryptanalysis

Affine ciphers can easily be broken by exhaustive key search (EKS), also known as brute force search, which consists of trying every possible key until the right one is found:

$$\text{there are } \phi(n) \times n \text{ possible keys}$$

As n is the size of the alphabet, n and $\phi(n)$ are usually small. For instance, $|V| = 26$ for the English alphabet of the small letters. This leads to $\phi(26) \times 26 = 12 \times 26 = 312$ keys.

Question: How do we know when we have found the correct plaintext?

Answer: We know that because it looks like a V-language message, or like a data file from a computer application, or like a database in a reasonable format; it looks like something understandable (in some way).

When we look at a cryptotext file, or a file decrypted with a wrong key, it looks like gibberish.

Vigenère cipher

This is a poly-alphabetic substitution cipher originally described by Giovan Battista Bellaso in 1553 [3], but later mis-attributed to Blaise de Vigenère (1523–1596).

Cipher 2 (Vigenère)

- $\mathcal{M} = \mathcal{C} = V^+$, *where* $V = \{v_0, \ldots, v_{n-1}\}$ *is some given ordered alphabet with* n *letters;*
- $\mathcal{K} = \mathbb{Z}_n^m$, *where* $m > 0$;
- *For any key* $K = (k_1, \ldots, k_m)$ *and* $v_{i_1} \cdots v_{i_t} \in V^+$,
  - $e_K(v_{i_1} \cdots v_{i_t}) = v_{i_1 \oplus k_1} \cdots v_{i_m \oplus k_m} v_{i_{m+1} \oplus k_1} \cdots$
  - $d_K(v_{i_1} \cdots v_{i_t}) = v_{i_1 \ominus k_1} \cdots v_{i_m \ominus k_m} v_{i_{m+1} \ominus k_1} \cdots$

  $\oplus$ *and* $\ominus$ *are the addition and subtraction modulo* n*, respectively.*

Vigenère cipher

Example 5

Assume V is the English small letters alphabet, $|V| = 26$. Let $m = 6$ and $K = (2, 8, 15, 7, 4, 17)$. Then

message	c	r	y	p	t	o	g	r	a	p	h
index	2	17	24	15	19	14	6	17	0	15	7
key	2	8	15	7	4	17	2	8	15	7	4
new index	4	25	13	22	23	5	8	25	15	22	11
ciphertext	e	z	n	w	x	f	i	z	p	w	l

message	c	r	y	p	t	o	g	r	a	p	h
key	C	I	P	H	E	R	C	I	P	H	E
ciphertext	e	z	n	w	x	f	i	z	p	w	l

(in the second table, we wrote the key in capital letters for visibility).

Vigenère cipher: cryptanalysis

Question: Does EKS work for Vigenère cipher?

Answer: There are n^m keys, and this number is large for large values of m. For instance, $26^{17} \approx 2^{4.7 \times 17} \approx 2^{80}$

Rule of thumb: feasible to only run a computer on a problem which takes under 2^{80} steps !

An attack against the Vigenère cipher should purport two phases:

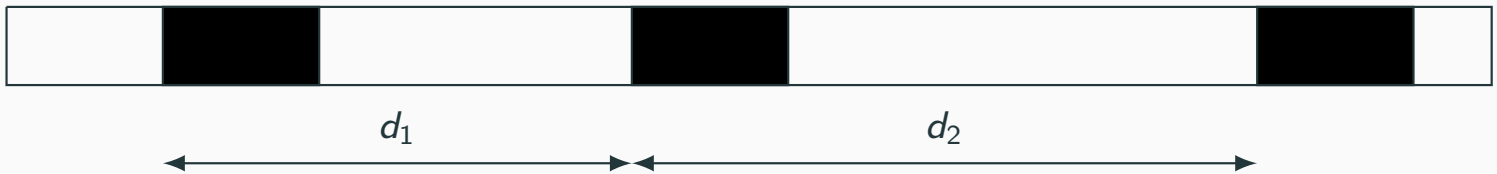
1. Finding the key length m:
   - Kasiski test;
   - Index of coincidence test;
2. Finding the key:
   - Mutual index of coincidence test;
   - Chi-squared (χ^2) test.

Vigenère cipher: Kasiski test

Proposed by F. W. Kasiski in 1863 [6].

Main idea: The distances between consecutive occurrences of the same string in the cryptotext are likely to be multiples of the key length m.

ciphertext with identical blocks



Look for m among the divisors of (d_1, d_2)

Vigenère cipher: Kasiski test

Example 6

ORLNHTWJDPBZGURWXSZRKIHTCZZTVOWJCLXRKULDMORGPDIFKRRTEMNVYGGT
QIRWXGCBCDYQQIPTVHPVQHMVGICLTGPFNXMMHFPJLVQNCKXFYVUXEZUKSSRA
QICSNFKEEIAWUJCBXGVVPIHDCJQIASGOYB

QI has four occurrences at 61, 73, 121 and 147. The distances between them are 12, 48, and 26. These numbers have only 1 and 2 as common divisors. An encryption key is unlikely to have a length of 1 or 2. As a result, we will look for divisors for 12 and 48. These are 1, 2, 3, 4, 6, and 12. According to the previous remark, we might expect a key of length 3, 4, 6, or 12.

Vigenère cipher: index of coincidence

The index of coincidence test, proposed by Friedman in 1922 [4], is based on the statistics of the underlying language (as an example, we will consider the case of the English language):

1. First order statistics: distribution of the underlying language letters (usually called letter frequencies)
2. Second order statistics: distribution of the most common sequences of two (digrams) or three (trigrams) letters of the underlying language

Example 7

Let p_i the frequency of the i-th letter. Many sources give $p_0 = 0.082$, $p_1 = 0.015$, $p_2 = 0.028$ and so on.

For digrams and trigrams, many sources give $p(\texttt{th}) = 0.0315$ or $p(\texttt{an})$=0.0172. As with respect to trigrams, `the` is the most frequent.

Vigenère cipher: index of coincidence

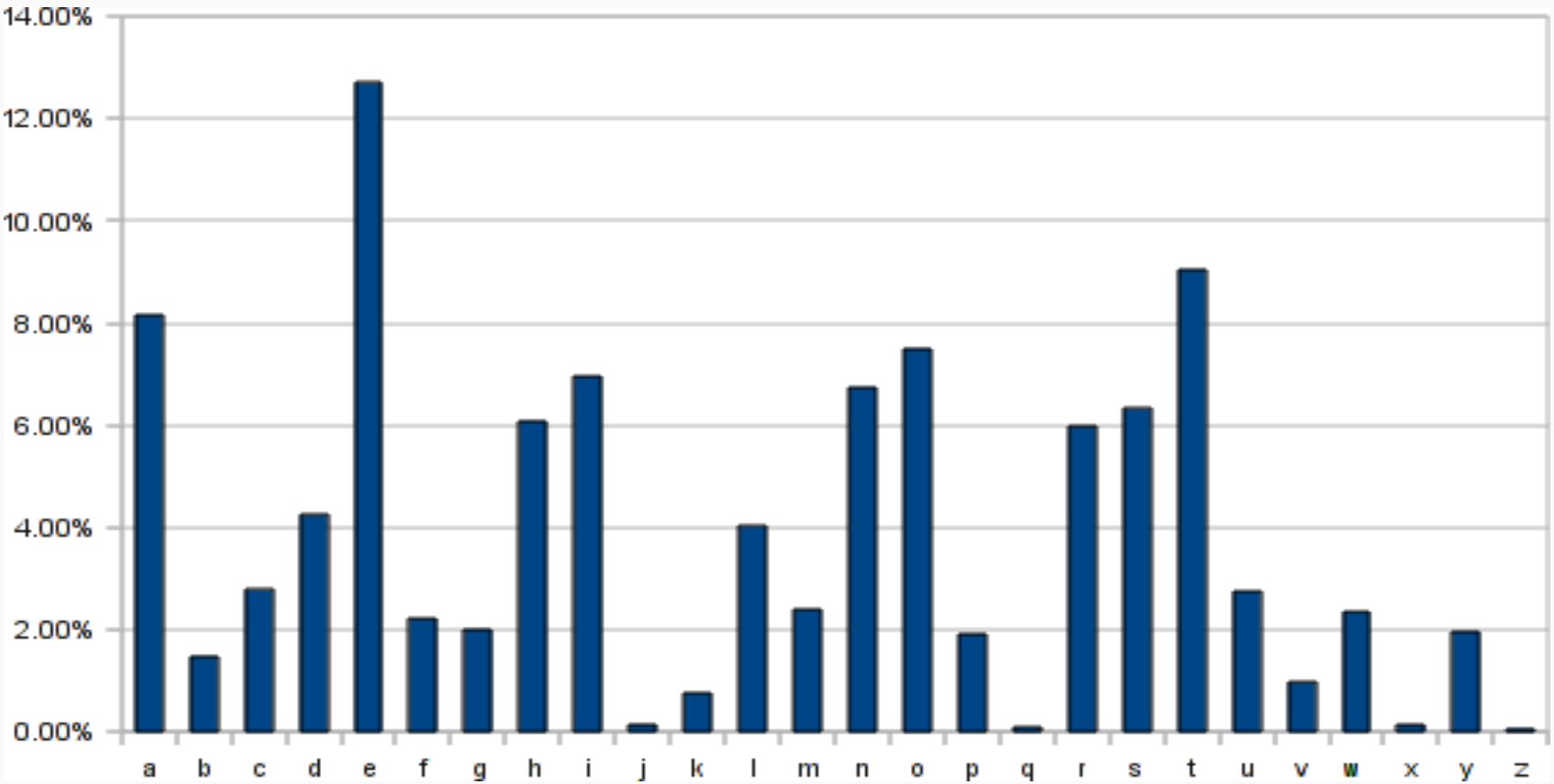


Figure 1: English letter frequency

Vigenère cipher: index of coincidence

The index of coincidence IC of a text x = probability that x has two occurrences of the same letter:

$$IC(x) = \sum_{i=0}^{25} \frac{f_i(x)}{|x|} \cdot \frac{f_i(x) - 1}{|x| - 1}$$

where $f_i(x)$ is the frequency of the i-th letter in x.

Example 8

1. "at the department of computer science in iasi we all are learning cryptography" – 0.06061 (spaces are not counted). IC does not change if we use uppercase letters;
2. "At the Department of Computer Science in Iasi, we all are learning cryptography." – 0.05092 (spaces are not counted).

Vigenère cipher: basic facts about *IC*

1. The expected average value for *IC* can be approximated by $\sum_{i=0}^{25} p_i^2$, and so it depends on the language in which the text is written

   English ~ 0.0667; French ~ 0.0778; German ~ 0.0762
   Spanish ~ 0.0770; Italian ~ 0.0738; Russian ~ 0.0529

2. $IC(x) = IC(x')$ for any x' obtained by shifting x's letters, because

$$f_i(x') = f_{i \ominus k}(x), \quad \text{for all } k \in \mathbb{Z}_{26};$$

3. If *IC* is high (close to 0.0667 for English), then the message is a plaintext or has probably been encrypted using a transposition or a mono-alphabetic substitution cipher;

4. If *IC* is low (close to 0.0385 for English), then the message has probably been encrypted using a poly-alphabetic cipher.

Vigenère cipher: finding the key length by *IC*

For a given string y and $m \geq 1$, let y_i be the order i substring of y

$$y_i = y(i)y(m+i)y(2m+i)\cdots$$

Remark 9

If $i \leq m$ and y is a Vigenère cryptotext, then y_i is obtained by shifting some source text x_i by the same shift k_i. Thus, $IC(y_i) = IC(x_i)$ and, as a conclusion, $IC(y_i)$ must be high.

Attack 1 (Finding the key length by *IC*)

Given a Vigenère cryptotext y, find the least $m \geq 1$ such that $IC(y_1), \ldots, IC(y_m)$ are closest to 0.0667.

Vigenère cipher: index of coincidence

Example 10 (Example 6 continued)

m	*index of coincidence*
1	0.03574;
2	0.04375 ± 0.031;
3	0.04186 ± 0.033;
4	0.04563 ± 0.029;
5	0.03669 ± 0.038;
6	0.06192 ± 0.016 ;
7	0.03278 ± 0.042.

We are led to assume that $m = 6$ might be the length of the key.

Vigenère cipher: mutual index of coincidence

The mutual index of coincidence (MIC) of two strings x and y [4] is probability that the two strings have a common letter:

$$MIC(x, y) = \sum_{i=0}^{25} \frac{f_i(x)}{|x|} \cdot \frac{f_i(y)}{|y|}.$$

Remark 11

Given a message x, a cryptotext y of x, a key length m, and $1 \leq j \leq m$, we expect

$$MIC(x, y_j) \approx \sum_{i=0}^{25} p_i \cdot \frac{f_i(y_j)}{|y_j|}.$$

Vigenère cipher: mutual index of coincidence

Remark 12

Using the notation in the remark above, if y_j is shifted by $\ell = -k_j \bmod n$ positions, denoted $y_j[\ell]$, then we expect

$$MIC(x, y_j[\ell]) \approx \sum_{i=0}^{25} p_i \cdot \frac{f_i(x_j)}{|x_j|} \approx \sum_{i=0}^{25} p_i^2.$$

It turns out that for $\ell \neq -k_j \bmod 26$ we have

$$MIC(x, y_j[\ell]) < 0.045.$$

Vigenère cipher: mutual index of coincidence

Attack 2 (ciphertext y and key length m)

For each $1 \leq j \leq m$ do:

1. *Compute ℓ such that the following value is maximum*

$$\sum_{i=0}^{25} p_i \cdot \frac{f_i(y_j[\ell])}{|y_j[\ell]|};$$

2. *Set $k_j = -\ell \bmod 26$.*

Example 13 (Example 10 continued)

If we apply the above algorithm to Example 10, we obtain $k_1 = 2$, $k_2 = 17$, $k_3 = 24$, $k_4 = 15$, $k_5 = 19$, and $k_6 = 14$. Therefore, the key is $K = CRYPTO$. The message is (spaces were added):

many of us failed the exam to fai because we did not appropriately learn. for the sake of correctness there was no pity for us however we will study more during this semester to pass the exam

Principles of modern cryptography

Letters and digits vs. binary bit sequences

1. Classical cryptography mainly uses traditional characters (i.e. letters and digits) directly;

2. Modern cryptography operates on binary bit sequences (as binary encoding of characters). This is closely related to the development of powerful computers that can perform binary operations on blocks of data at a time instead of individual characters.

“Security through obscurity” vs. “open design”

1. Classical cryptography treats the security issue of ciphers through obscurity:

   > Security through obscurity is the reliance on the secrecy of the implementation of a system or components of a system to keep it secure

2. Modern cryptography relies on Auguste Kerckhoffs’ principle formulated in 1883 [8, 9]:

   > A cipher should be secure even if everything about the system, except the key, is public knowledge

   Advantages of adopting this principle:

   2.1 It is much easier for the parties to keep secret a short key than the entire algorithm;
   2.2 If the key is exposed, it is much easier to change the key than the entire algorithm.

“Security through obscurity” vs. “security”

1. Modern cryptography comes with formal definitions of security together with rigorous proofs of security for cryptographic primitives

   Murphy’s law: If there is a single security hole, someone will eventually find it

2. Modern cryptography takes into account future technologies. As the secret keys can be tried exhaustively and the computers become faster and faster, secret keys should be long enough to resist exhaustive search

   Moore’s law: the speed of CPUs doubles every 18 months

From military purposes to society

1. Classical cryptography was mainly devoted to encryption for governmental and military purposes;
2. Modern cryptography focuses on three fundamental paradigms:
   - 2.1 Confidentiality;
   - 2.2 Integrity;
   - 2.3 Authentication;
3. Modern cryptography answers dedicated problems arrived from the modern society:
   - 3.1 Electronic commerce;
   - 3.2 Electronic voting.

   Modern cryptography is everywhere now.

Reading and exercise guide

Reading and exercise guide

Course readings:

1. Pages 1-22 from [7].

The history of cryptography is fascinating. Try to discover this through three of the best books on the history of cryptography [5, 2, 1].

References

[1] Bauer, C. (2021). *Secret History: The Story of Cryptology*. Chapman & Hall/CRC Cryptography and network security series. CRC Press, second edition.

[2] Bauer, F. L. (2006). *Decrypted Secrets. Methods and Maxims of Cryptology*. Springer, 4th edition.

[3] Bellaso, G. B. (1553). *La cifra del sig. Giovan Battista Belaso*. Venice.

[4] Friedman, W. F. (1922). *The Index of Coincidence and Its Applications in Cryptography*. Number 49 in Cryptographic Series. Riverbank Laboratories.

[5] Kahn, D. (1996). *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, rev sub edition.

[6] Kasiski, F. (1863). *Die Geheimschriften und die Dechiffrirkunst*. Mittler & Son.

[7] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

[8] Kerckhoffs, A. (1883a). La cryptographie militaire. *Journal des sciences militaires*, IX.

[9] Kerckhoffs, A. (1883b). La cryptographie militaire. *Journal des sciences militaires*, IX.

[10] Shannon, C. (1949). Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715.

Perfect Security

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Perfect security

One-time Pad

Equivalent formulations of perfect security

Limitations of perfect security

Reading and exercise guide

Perfect security

Security property and adversary

A cryptographic scheme has the role of protecting a system by providing it with a certain security property!

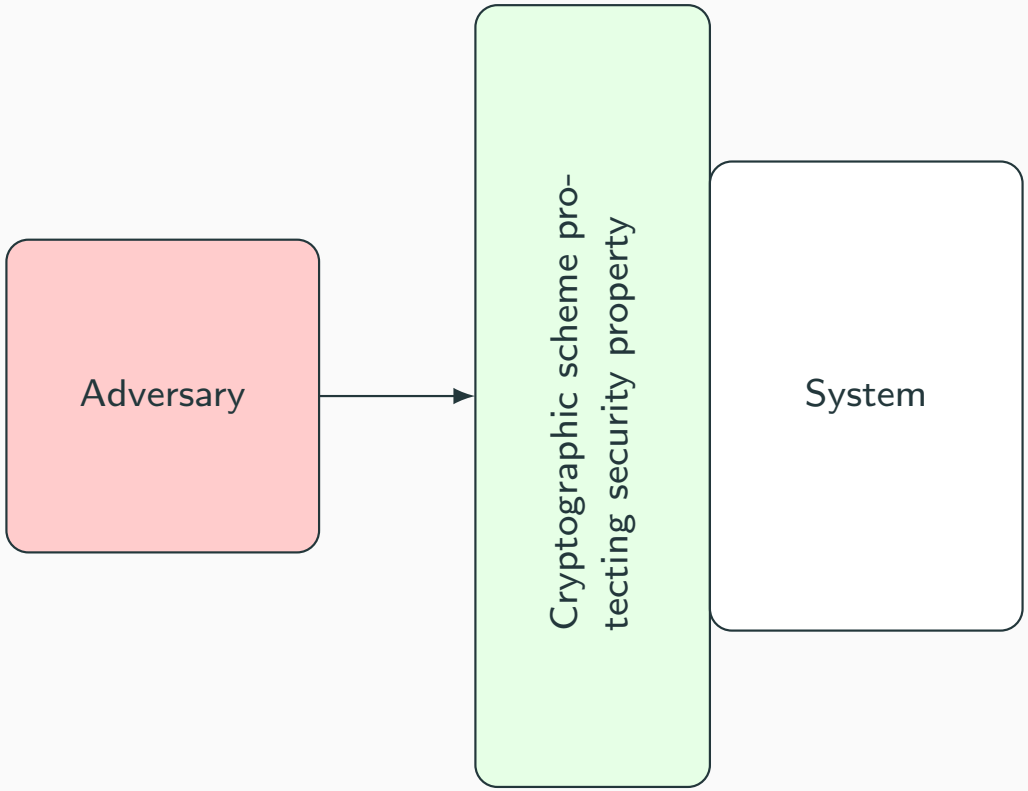


Figure 1: Adversary trying to penetrate the cryptographic scheme protecting the system

Security model

Security goal

Security property

1. Confidentiality (secrecy);
2. Integrity;
3. (Unilateral, mutual, data origin, etc.) authentication;
4. Collision resistance;
5. One-wayness;
6. etc.

Attack model

Adversary type

1. Unlimited computational power (e.g. Dolev-Yao adversary);
2. Restricted computational power such as PPT algorithm;

Attack type

1. Passive;
2. Active;
3. etc.

Security model = security goal + attack model

What does perfect security mean?

- Introduced first by Claude Shannon in 1949 [3];
- It was the first treatment of the security problem (for ciphers);
- Perfect security = information-theoretic security = unconditional security for ciphers;
- Security model:
  - security goal: the ciphertext reveals no information about the plaintext;
  - attack model: adversary with unlimited power, under the ciphertext-only attack;
- Formalization: by conditional probabilities (or entropy).

Probability distributions associated to ciphers

Notations and assumption regarding ciphers $\mathcal{S} = (\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{E}, \mathcal{D})$:

1. $P_{\mathcal{K}}$ stands for the probability distribution on $\mathcal{K}$;
2. $P_{\mathcal{M}}$ stands for the probability distribution on $\mathcal{M}$;
3. Assume $P_{\mathcal{K}}(K) > 0$ and $P_{\mathcal{M}}(m) > 0$, for all $K \in \mathcal{K}$ and $m \in \mathcal{M}$;
4. For any $c \in \mathcal{C}$ there exist $m \in \mathcal{M}$ and $K \in \mathcal{K}$ such that $e_K(m) = c$;
5. Assume that the random variables induced by the two probability distributions are independent.

The fourth assumption is only technical to help defining conditional probabilities (see next slide) – if avoided, its leads to tedious analysis cases.

Probability distribution on ciphertexts

$P_{\mathcal{K}}$ and $P_{\mathcal{M}}$ give rise to a probability distribution $P_{\mathcal{C}}$ on $\mathcal{C}$:

$$P_{\mathcal{C}}(c) = \sum_{(K,m)\in A_c} P_{\mathcal{K}}(K)P_{\mathcal{M}}(m),$$

for all $c \in \mathcal{C}$, where $A_c = \{(K, m) \in \mathcal{K} \times \mathcal{M} \mid e_K(m) = c\}$.

According to our assumptions, $P_{\mathcal{C}}(c) > 0$, for all c. Therefore, for all $m \in \mathcal{M}$ and $c \in \mathcal{C}$, we may consider

$$P_{\mathcal{M}}(m \mid c) = \frac{P_{\mathcal{C}}(c \mid m)P_{\mathcal{M}}(m)}{P_{\mathcal{C}}(c)}.$$

Meaning of $P_{\mathcal{M}}(m \mid c)$: Probability that m is the plaintext when c is the ciphertext.

Probability distribution on ciphertexts: example

$\mathcal{M}$	a	b
	$1/4$	$3/4$

$\mathcal{K}$	K_1	K_2	K_3
	$1/2$	$1/4$	$1/4$

$\mathcal{E}$	a	b
K_1	1	2
K_2	2	3
K_3	3	4

Then,

$$\begin{aligned} P_{\mathcal{C}}(1) &= 1/8 \\ P_{\mathcal{C}}(2) &= 3/8 + 1/16 = 7/16 \\ P_{\mathcal{C}}(3) &= 3/16 + 1/16 = 1/4 \\ P_{\mathcal{C}}(4) &= 3/16 \end{aligned}$$

and

$$\begin{aligned} P_{\mathcal{M}}(a \mid 1) &= 1 & P_{\mathcal{M}}(b \mid 1) &= 0 \\ P_{\mathcal{M}}(a \mid 2) &= 1/7 & P_{\mathcal{M}}(b \mid 2) &= 6/7 \\ P_{\mathcal{M}}(a \mid 3) &= 1/4 & P_{\mathcal{M}}(b \mid 3) &= 3/4 \\ P_{\mathcal{M}}(a \mid 4) &= 0 & P_{\mathcal{M}}(b \mid 4) &= 1. \end{aligned}$$

Perfect security

Definition 1

A cipher $\mathcal{S}$ is perfectly secure if

$$P_{\mathcal{M}}(m \mid c) = P_{\mathcal{M}}(m)\,,$$

for all $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

Meaning of $P_{\mathcal{C}}(c \mid m)$: Probability that c is the ciphertext when m is the plaintext.

Theorem 2

A cipher $\mathcal{S}$ is perfectly secure iff $P_{\mathcal{C}}(c \mid m) = P_{\mathcal{C}}(c)$, for all $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

Definition 1 may be adopted for randomized ciphers too, and Theorem 2 holds in this case as well!

One-time Pad

One-time Pad (OTP)

- Frank Miller in 1882 wrote the following in the preface of his paper Telegraphic Code to Insure Privacy and Secrecy in the Transmission of Telegrams:

  > *"A banker in the West should prepare a list of irregular numbers to be called "shift-numbers," ... The difference between such numbers must not be regular. When a shift-number has been applied, or used, it must be erased from the list and not used again."*

- Gilbert Vernam proposed in 1917 (and patented in 1919) a teleprinter cipher in which a previously prepared key, kept on paper tape, is combined character by character with the plaintext message to produce the ciphertext
- Joseph Mauborgne "suggested" random keying material to Vernam's cipher (to make cryptanalysis impossible)

One-time Pad (OTP)

- According to several independent specialists in cryptography (including Steven Bellovin), Frank Miller was undoubtedly the first to propose the OTP concept
- According to David Kahn, Frank Miller was not conscious about his invention:

  > *"Miller probably invented the one-time pad, but without knowing why it was perfectly secure or even that it was ... Moreover, unlike Mauborgne's conscious invention, or the Germans' conscious adoption of the one-time pad to superencipher their Foreign Office codes, it had no echo, no use in cryptology. It sank without a trace ..."*

- Nowadays, the OTP is largely accredited to Vernam in 1917

One-time Pad (OTP)

Cipher 1 (One-time Pad (OTP))

- $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^\ell$, *where* $\ell \geq 1$
- *Keys are uniformly at random generated*
- *For any key* $K = (k_1, \ldots, k_\ell)$ *and* $m = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$,

$$e_K(m) = d_K(m) = (m_1 \oplus k_1, \ldots, m_\ell \oplus k_\ell)$$

Theorem 3

OTP is perfectly secure.

Proof.

Use Theorem 2. □

Equivalent formulations of perfect security

Perfect indistinguishability

Theorem 4 (Perfect indistinguishability)

A cipher $\mathcal{S}$ is perfectly secure iff $P_{\mathcal{C}}(c \mid m_0) = P_{\mathcal{C}}(c \mid m_1)$, *for all* $m_0, m_1 \in \mathcal{M}$ *and* $c \in \mathcal{C}$.

Proof.

Direct implication: from Theorem 2.

Converse implication: use $P_{\mathcal{C}}(c) = \sum_{m \in \mathcal{M}} P_{\mathcal{C}}(c \mid m) P_{\mathcal{M}}(m)$ and the hypothesis. □

Meanings:

- In a perfectly secure cipher, the probability distribution over $\mathcal{C}$ is independent of the plaintext;
- In a perfectly secure cipher, it is impossible to distinguish an encryption of m_0 from one of m_1 based on a given ciphertext.

Adversaries as probabilistic algorithms

Terminology and notation:

1. Adversary: probabilistic algorithm;
2. The invocation of a deterministic algorithm $\mathcal{A}$ on an input x with the output y is written as $y = \mathcal{A}(x)$ or $y := \mathcal{A}(x)$;
3. The invocation of a probabilistic algorithm $\mathcal{A}$ on an input x with the output y is written as $y \leftarrow \mathcal{A}(x)$;
4. The probability that the probabilistic algorithm $\mathcal{A}$ outputs y on the input x is denoted $P(y \leftarrow \mathcal{A}(x))$;
5. The probability that the probabilistic algorithm $\mathcal{A}$ outputs y is denoted $P(y \leftarrow \mathcal{A}(x) : x \leftarrow D)$, or $P(y \leftarrow \mathcal{A})$ when the input domain D with its probability distribution is clear from the context, and it is computed by

$$P(y \leftarrow \mathcal{A}(x) : x \leftarrow D) = \sum_{x \in D} P(x)P(y \leftarrow \mathcal{A}(x))$$

Algorithmic indistinguishability

Terminology and notation:

1. Given a cipher $\mathcal{S}$ and a message m, $\mathcal{E}(m)$ stands for the set of ciphertexts obtained by encrypting m. The probability distributions on $\mathcal{M}$ and $\mathcal{K}$ induce a probability distribution on $\mathcal{E}(m)$;
2. A probabilistic algorithm $\mathcal{A}$ that outputs m_0 or m_1 when inputs $m_0, m_1 \in \mathcal{M}$ and some $c \in \mathcal{C}$, is referred to as a distinguishing algorithm for $\mathcal{M}$.

Theorem 5 (Algorithmic indistinguishability)

A cipher $\mathcal{S}$ is perfectly secure iff

$$P(m_0 \leftarrow \mathcal{A}(m_0, m_1, c) : c \leftarrow \mathcal{E}(m_0)) = P(m_0 \leftarrow \mathcal{A}(m_0, m_1, c) : c \leftarrow \mathcal{E}(m_1))$$

for any distinguishing algorithm $\mathcal{A}$ for $\mathcal{M}$ and any $m_0, m_1 \in \mathcal{M}$.

Algorithmic indistinguishability

Corollary 6

A cipher $\mathcal{S}$ is perfectly secure iff

$$P(m_b \leftarrow \mathcal{A}(m_0, m_1, c) : b \leftarrow \{0,1\},\ c \leftarrow \mathcal{E}(m_b)) = \frac{1}{2}$$

for any distinguishing algorithm $\mathcal{A}$ for $\mathcal{M}$ and any $m_0, m_1 \in \mathcal{M}$ with $m_0 \neq m_1$.

Corollary 6 characterizes a perfectly secure encryption scheme under an eavesdropper $\mathcal{A}$.

Corollary 6 holds true even if $\mathcal{E}$ is a randomized algorithm or $\mathcal{A}$ would use encryption or decryption oracles !

Limitations of perfect security

Direct consequences of perfect security

Remark 7

If $\mathcal{S}$ is a perfectly secure cipher, then

$$(\forall m \in \mathcal{M})(\forall c \in \mathcal{C})(\exists K \in \mathcal{K})(e_K(m) = c)\,.$$

Indeed, given $m \in \mathcal{M}$ and $c \in \mathcal{C}$, the perfect security of $\mathcal{S}$ leads to

$$P_{\mathcal{C}}(c \mid m) = P_{\mathcal{C}}(c).$$

As $P_{\mathcal{C}}(c) > 0$, it follows $P_{\mathcal{C}}(c \mid m) > 0$. Therefore,

$$P_{\mathcal{C}}(c \mid m) = \sum_{K \in \mathcal{K}:\ e_K(m)=c} P_{\mathcal{K}}(K) > 0,$$

which means that there exists at least one key K with $e_K(m) = c$.

This property holds true even if $\mathcal{E}$ is a randomized algorithm!

Direct consequences of perfect security

Remark 8

If $\mathcal{S}$ is a perfectly secure cipher, then $|\mathcal{K}| \geq |\mathcal{M}|$ *. Indeed, given a ciphertext c, we have*

$$\mathcal{M} = \{d_K(c) \mid K \in \mathcal{K}\}$$

in the view of Remark 7. Therefore, $|\mathcal{K}| \geq |\mathcal{M}|$.

Remark 9

If $\mathcal{S}$ is a perfectly secure cipher, then $|\mathcal{K}| \geq |\mathcal{C}|$ *. Indeed, given a message $m \in \mathcal{M}$, we have*

$$\mathcal{C} = \{e_K(m) \mid K \in \mathcal{K}\}$$

in the view of Remark 7. Therefore, $|\mathcal{K}| \geq |\mathcal{C}|$.

These properties hold true even if $\mathcal{E}$ is a randomized algorithm!

Shannon's theorem

Theorem 10 (Shannon's theorem)

Let $\mathcal{S}$ be a cipher and $P_{\mathcal{K}}$ and $P_{\mathcal{M}}$ be the probability distributions on $\mathcal{K}$ and $\mathcal{M}$, respectively. If:

1. *For any $m \in \mathcal{M}$ and $c \in \mathcal{C}$ there exists unique $K \in \mathcal{K}$ such that $e_K(m) = c$, and*
2. *$P_{\mathcal{K}}$ is the uniform distribution on $\mathcal{K}$,*

then $\mathcal{S}$ is a perfectly secure cipher.

Conversely, if $\mathcal{S}$ is perfectly secure and $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$, then the items (1) and (2) above hold.

Reading and exercise guide

Reading and exercise guide

Course readings:

1. Pages 23-36 from [2].

I also recommend you [1], where you can find a gently introduction to RFID security models and PUFs.

References

[1] Țiplea, F. L., Andriesei, C., and Hristea, C. (2021). Security and privacy of PUF-based RFID systems. In Bernardini, R., editor, *Cryptography*, chapter 5. IntechOpen, Rijeka.

[2] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

[3] Shannon, C. (1949). Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715.

Indistinguishability

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Introduction

Introduction

Indistinguishability and pseudo-randomness are cornerstones of modern cryptography!

Indistinguishability and pseudo-randomness are crucial in defining security concepts for many cryptographic primitives!

Negligible functions

Negligible functions

A function is negligible if it is smaller than the inverse of any polynomial.

Definition 1

$f : \mathbb{N} \to \mathbb{R}$ is called negligible if for any real constant $c > 0$ there exists $n_0 > 0$ such that $|f(n)| < 1/n^c$, for all $n \geq n_0$.

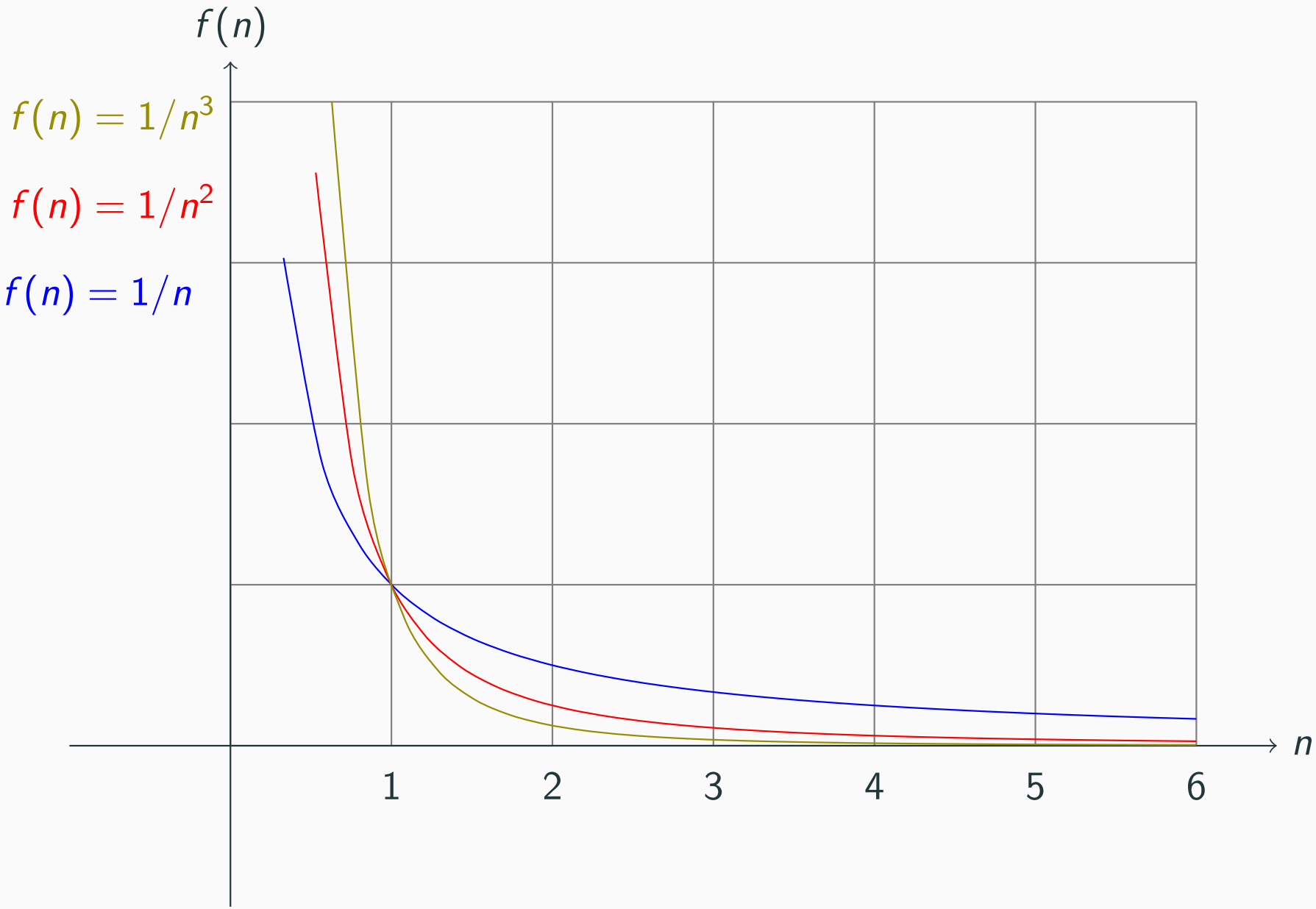
Example 2

1. Negligible functions: 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$
2. Non-negligible functions: n^{-100}, $2^{-c \log n}$

Theorem 3

A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if and only if $\lim_{n\to\infty} f(n) \cdot n^c = 0$, for all $c > 0$.

Negligible functions



Super-poly and poly-bounded functions

A function is super-poly it it is larger than any polynomial, and it is poly-bounded if it is bounded by some polynomial.

Definition 4

Let $f : \mathbb{N} \to \mathbb{R}$ be a function.

1. f is called super-poly if $1/f$ is negligible;
2. f is called poly-bounded if $| f(n) | \leq n^c + d$ for some real constants $c, d \geq 0$ and all n.

Example 5

Let $f : \mathbb{N} \to \mathbb{R}$ be the function given by

$$f(n) = \begin{cases} 1/n, & \text{if } n \text{ is even} \\ 1/2^n, & \text{otherwise} \end{cases}$$

f is non-negligible and $1/f$ is neither super-poly nor poly-bounded.

Statistical indistinguishability

Statistical distance

Definition 6

Let X and Y be two random variables with range V. The statistical distance (or total variation distance) between X and Y, denoted $\Delta(X, Y)$, is defined as

$$\Delta(X, Y) = \frac{1}{2} \sum_{v \in V} |P(X = v) - P(Y = v)|$$

Remarks and terminology:

- X and Y may have different ranges, V_1 and V_2, resp. In this case we take $V = V_1 \cup V_2$ and extend X and Y with zero probabilities for the missing values
- X and Y are ϵ-close is $\Delta(X, Y) \leq \epsilon$

Graphical view of statistical distance

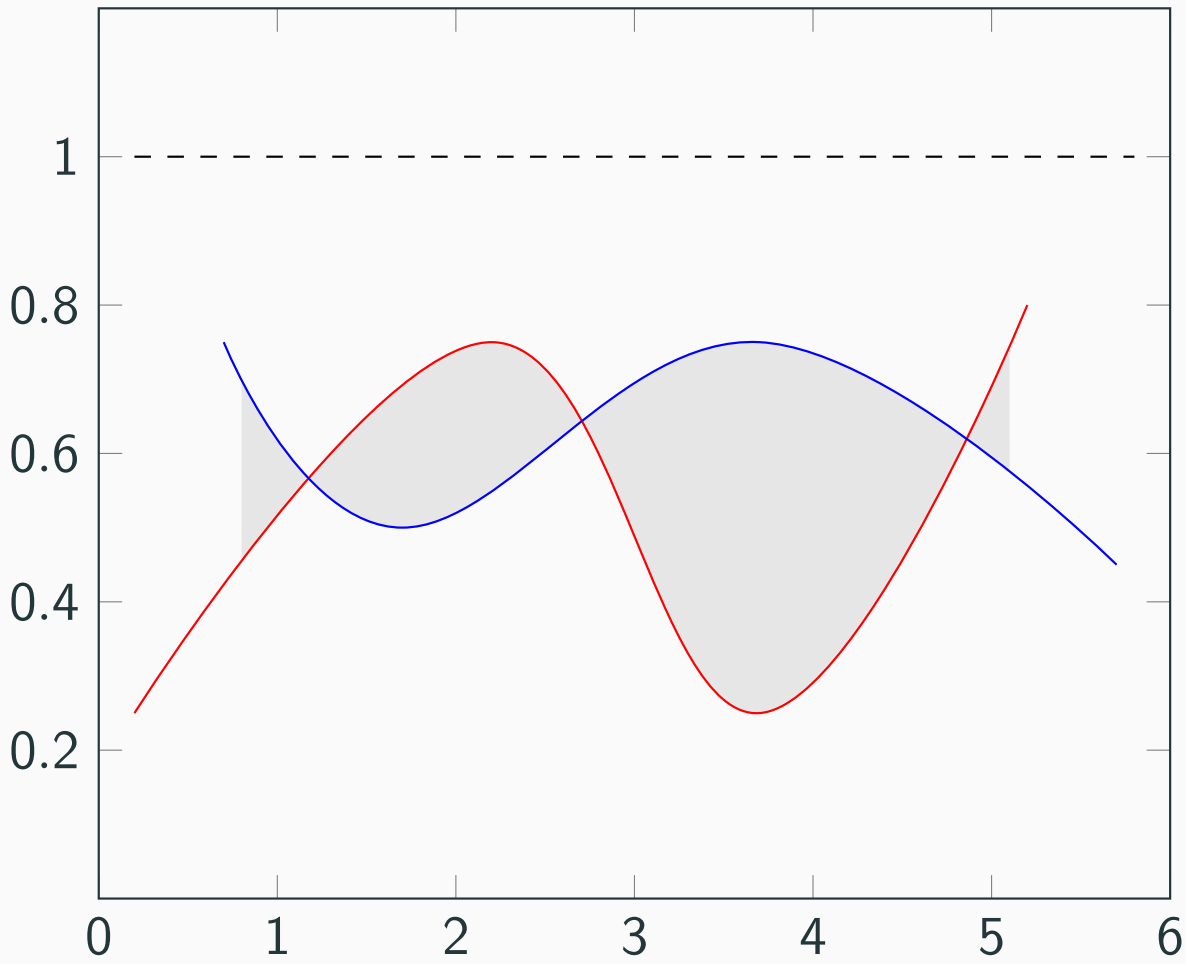


Figure 1: Two random variables and their statistical distance in the interval $[0.8, 5.1]$

Statistical distance: example

Example 7

Assume:

- A is an integer with $2^n \leq A < 2^{n+1}$, for some $n \geq 1$
- X is a random variable that takes integer values v in $[0, A)$ with probabilities $1/A$
- Y is a random variable that takes integer values v in $[0, A)$ with probabilities $1/2^n$, for all $v < 2^n$, and 0, otherwise.

Then,

$$\Delta(X, Y) = \frac{A - 2^n}{A}$$

When A approaches 2^n, $\Delta(X, Y)$ approaches 0.

Terminology on random variables

Let X and Y be random variables with range V:

- X and Y are identical, denoted $X \equiv Y$, if $P(X = v) = P(Y = v)$ for all $v \in V$
- X and Y are disjoint if, for all $v \in V$, either $P(X = v) = 0$ or $P(Y = v) = 0$ (X and Y do not output common values)
- If $f : V \to V'$ is a function, then $f(X)$ is a random variable with range V' and

$$P(f(X) = v') = P(X \in f^{-1}(\{v'\}))$$

- If $\mathcal{A}$ is a probabilistic algorithm (which inputs values from V), then $\mathcal{A}(X)$ is the random variable
  1. Sample $x \leftarrow X$;
  2. Sample $a \leftarrow \mathcal{A}(x)$;
  3. Output a.

Statistical distance: basic properties

Theorem 8

Let X, Y, and Z be random variables with range V. Then,

1. $\Delta(X, Y) \in [0, 1]$;
2. $\Delta(X, Y) = 0$ *iff* X *and* Y *are identical;*
3. $\Delta(X, Y) = 1$ *iff* X *and* Y *are disjoint;*
4. $\Delta(X, Y) = \Delta(Y, X)$;
5. $\Delta(X, Z) \leq \Delta(X, Y) + \Delta(Y, Z)$.

Theorem 9

Let X and Y be two random variables with range V and $\mathcal{A}$ be a probabilistic algorithm. Then,

$$\Delta(\mathcal{A}(X), \mathcal{A}(Y)) \leq \Delta(X, Y)$$

Statistical indistinguishability

Notation:

- Family of random variables: $X = (X_n)_{n\in\mathbb{N}}$

Definition 10

Two families of random variables $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (Y_n)_{n\in\mathbb{N}}$ are called statistically indistinguishable, denoted $X \approx_s Y$, if the function $\Delta(n) = \Delta(X_n, Y_n)$ is negligible as a function of n.

Definition 11

Two families of random variables $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (Y_n)_{n\in\mathbb{N}}$ are called perfect indistinguishable, denoted $X \approx Y$, if the function $\Delta(n) = \Delta(X_n, Y_n)$ is zero as a function of n.

Statistical indistinguishability: example

Example 12

Assume X_n outputs uniformly at random vectors in $\{0,1\}^n$ and Y_n outputs uniformly at random vectors in $\{0,1\}^n \setminus \{0^n\}$, for all $n \geq 0$. Then, a simple computation leads to

$$\Delta(X_n, Y_n) = 2^{-n}$$

showing that $X = (X_n)_{n\geq 0}$ and $Y = (Y_n)_{n\geq 0}$ are statistically indistinguishable.

The conclusion of the example holds even if Y_n leaves out a negligible (as a function of n) portion of $\{0,1\}^n$!

Computational indistinguishability

Distinguisher

Definition 13

Let X and Y be random variables with range V. A distinguisher for X and Y is an algorithm $\mathcal{A}$ that, on an input from V outputs a bit.

One may think that $\mathcal{A}$ identifies X when outputting 0, and Y when outputting 1 (or vice-versa – it does not matter)

Definition 14

Let X and Y be random variables and $\mathcal{A}$ a distinguisher for them. The advantage of $\mathcal{A}$ in distinguishing X and Y, denoted $Adv_{\mathcal{A},X,Y}$, is defined as

$$Adv_{\mathcal{A},X,Y} = |P(1 \leftarrow \mathcal{A}(X)) - P(1 \leftarrow \mathcal{A}(Y))|$$

One may see that $Adv_{\mathcal{A},X,Y} = |P(0 \leftarrow \mathcal{A}(X)) - P(0 \leftarrow \mathcal{A}(Y))|$

Distinguisher

Theorem 15

Let X and Y be random variables. Then, for any distinguisher $\mathcal{A}$ for X and Y the following property holds:

$$Adv_{\mathcal{A},X,Y} = \Delta(\mathcal{A}(X), \mathcal{A}(Y))$$

Corollary 16

Let X, Y, and Z be random variables, and $\mathcal{A}$ be distinguisher. Then,

1. $Adv_{\mathcal{A},X,Y} \in [0,1]$;
2. $Adv_{\mathcal{A},X,Y} = 0$ *if* X *and* Y *are identical;*
3. $Adv_{\mathcal{A},X,Y} = Adv_{\mathcal{A},Y,X}$;
4. $Adv_{\mathcal{A},X,Z} \leq Adv_{\mathcal{A},X,Y} + Adv_{\mathcal{A},Y,Z}$;
5. $Adv_{\mathcal{A},X,Y} \leq \Delta(X, Y)$.

Computational indistinguishability

Definition 17

Two families of random variables $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (X_n)_{n\in\mathbb{N}}$ are called computationally indistinguishable, denoted $X \approx_c Y$, if $Adv_{\mathcal{A},X,Y}(n)$ is negligible as a function of n, for all PPT distinguishers $\mathcal{A}$ for X and Y.

Definition 18

1. A distinguisher $\mathcal{A}$ distinguishes the random variables X and Y with probability ϵ, if $Adv_{\mathcal{A},X,Y} > \epsilon$.
2. A distinguisher $\mathcal{A}$ distinguishes the families of random variables $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (X_n)_{n\in\mathbb{N}}$ with probability $\epsilon(n)$, if $\mathcal{A}$ distinguishes X_n and Y_n with probability $\epsilon(n)$, for all $n \in \mathbb{N}$.

Computational indistinguishability: basic properties

Theorem 19

1. $\approx_c$ *is reflexive, symmetric, and transitive.*
2. *If* $X \approx_s Y$ *then* $X \approx_c Y$.
3. *Closure under PPT algorithms: If* $X \approx_c Y$ *then* $\mathcal{A}(X) \approx_c \mathcal{A}(Y)$, *for any PPT algorithm* $\mathcal{A}$.

A generalization of transitivity:

Lemma 20 (Hybrid Lemma)

Let $X^1, \ldots, X^m$ *be families of random variables and* $\mathcal{A}$ *be a distinguisher that distinguishes* X^1 *and* X^m *with probability* $\epsilon(n)$, *where* $m \geq 2$. *Then, there exists* i *such that* $\mathcal{A}$ *distinguishes* X^i *and* X^{i+1} *with probability* $\frac{\epsilon(n)}{m}$.

Computational indistinguishability and unpredictability

1. If two distribution are computationally indistinguishable then they are unpredictable: no efficient algorithm can predict which distribution a sample comes from with probability significantly better than $1/2$;

2. If it is not possible to predict which distribution a sample comes from with a probability significantly better than $1/2$, then the distributions must be computationally indistinguishable.

Reading and exercise guide

Reading and exercise guide

Course readings:

1. Pages 45-49 and 296-298 from [2].

For a comprehensive treatment of the field, I recommend Chapter 3 of [1].

References

[1] Goldreich, O. (2004). *Foundations of Cryptography. Basic Tools*. Cambridge University Press, first edition.

[2] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

Pseudo-random Generators

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Pseudo-randomness

Pseudo-random generators

Pseudo-random generators with one bit expansion

Pseudo-random generators with polynomial expansion

PRGs in practice

Reading and exercise guide

Pseudo-randomness

Notation on probability distributions

Definition 1

A probability ensemble [3], also called family of random variables, is an indexed set of random variables $X = (X_i)_{i \in \mathcal{I}}$, where $\mathcal{I}$ is at most countable.

In cryptography, probability ensembles are also commonly referred to as probability distributions.

Typically in our applications:

- $\mathcal{I} = \mathbb{N}$;
- X_n ranges over $\{0,1\}^{\ell(n)}$, for some polynomial ℓ with positive values, and for all n. To highlight ℓ, we will write sometimes

$$X_\ell = (X_{\ell,n})_{n \in \mathbb{N}}$$

Notation on probability distributions

The uniform distribution is denoted

$$U_\ell = (U_{\ell,n})_{n\in\mathbb{N}}$$

where $U_{\ell,n}$ is the uniform distribution over $\{0,1\}^{\ell(n)}$

Other notations and conventions:

1. When $\ell(n) = n$ for all n, U_ℓ is simply denoted by U;
2. Given $t \in \{0,1\}^n$,
   - t_i stands for the ith bit of t, $1 \leq i \leq n$;
   - $t[1,i]$ stands for the prefix $t_1 \cdots t_i$.

The subscript ℓ must not be confused with the index that refers to the ℓ-th component of some family of distributions!

Pseudo-random distributions

Definition 2

A family of distributions $X_\ell = (X_{\ell,n})_{n\in\mathbb{N}}$ is called pseudo-random if $X_\ell \approx_c U_\ell$, that is

$$|P(1 \leftarrow \mathcal{A}(X_{\ell,n})) - P(1 \leftarrow \mathcal{A}(U_{\ell,n}))| \tag{1}$$

is negligible (as a function of n), for any PPT distinguisher $\mathcal{A}$ for X_ℓ and U_ℓ.

Terminology:

- A PPT algorithm $\mathcal{A}$ as in the above definition is called a polynomial time statistical test.
- If (1) above fails we say that X_ℓ fails the test $\mathcal{A}$.

The next bit test

Andrew C.-C. Yao: Pólya Prize (1987), Knuth Prize (1996), Turing Award (2000)

Definition 3

$X_\ell = (X_{\ell,n})_{n\in\mathbb{N}}$ passes the next bit test if for any PPT algorithm $\mathcal{A}$ and $1 \leq i < \ell(n)$,

$$\left| P(t_{i+1} \leftarrow \mathcal{A}(1^n, t[1,i]) \mid t \leftarrow X_{\ell,n}) - \frac{1}{2} \right|$$

is negligible (as a function of n).

Theorem 4 (Yao, 1982)

A family of distributions $X_\ell = (X_{\ell,n})_{n\in\mathbb{N}}$ is pseudo-random if and only if it passes the next bit test.

Pseudo-random generators

Pseudo-random generator

Definition 5

A pseudo-random generator (PRG) is a deterministic polynomial-time algorithm G satisfying the following conditions:

1. There exists a function $\ell : \mathbb{N} \to \mathbb{N}$ such that $\ell(n) > n$ for all n;
2. $|G(s)| = \ell(|s|)$ for all $s \in \{0,1\}^*$;
3. The distribution $(G_{\ell,n})_{n\in\mathbb{N}} = (G(U_n))_{n\in\mathbb{N}}$ induced by G is pseudo-random.

Terminology:

- The function ℓ is called the stretch or the expansion factor of the generator.
- The input s to the generator is called seed.

Generators that are not pseudo-random

Example 6

$$G(x_1 \cdots x_n) = \begin{cases} x_1 \cdots x_n x_1, & \text{if } n > 0 \\ 0, & \text{otherwise,} \end{cases}$$

for all $n \geq 0$ and $x_1 \cdots x_n \in \{0,1\}^n$ (in this case, $\ell(n) = n + 1$).

G is not pseudo-random. To see that, consider the PPT algorithm $\mathcal{A}$ given by

$$1 \leftarrow \mathcal{A}(y_1 \cdots y_n y_{n+1}) \quad \Leftrightarrow \quad y_1 = y_{n+1}$$

for all $n > 0$ and $y \in \{0,1\}^{n+1}$. Then,

$$P(1 \leftarrow \mathcal{A}(G_{\ell,n})) = 1$$

and

$$P(1 \leftarrow \mathcal{A}(U_{\ell,n})) = 1/2,$$

from which one can easily deduce that G is not pseudo-random.

Generators that are not pseudo-random

Example 7

$$G(x_1 \cdots x_n) = \begin{cases} x_1(x_1 \oplus x_2) \cdots (x_{n-1} \oplus x_n)x_n, & \text{if } n > 0 \\ 0, & \text{otherwise,} \end{cases}$$

for all $n \geq 0$ and $x_1 \cdots x_n \in \{0,1\}^n$ (in this case, $\ell(n) = n + 1$).

G is not pseudo-random. To see that, consider the PPT algorithm $\mathcal{A}$:

1. on an input $x_1 y_2 \cdots y_n y_{n+1}$, decompose it as follows

$$\begin{aligned} y_2 &= x_1 \oplus x_2 \\ y_3 &= x_2 \oplus x_3 \\ &\cdots \\ y_n &= x_{n-1} \oplus x_n \end{aligned}$$

2. output 1 if and only if $x_n = y_{n+1}$.

One may calculate then the same probabilities as in Example 6.

Generators that are not pseudo-random

Example 8

$$G(x_1 \cdots x_n) = \begin{cases} x_1 \cdots x_n(x_1 \oplus \cdots \oplus x_n), & \text{if } n > 0 \\ 0, & \text{otherwise,} \end{cases}$$

for all $n \geq 0$ and $x_1 \cdots x_n \in \{0,1\}^n$ (in this case, $\ell(n) = n + 1$).

G is not pseudo-random. To see that, consider the PPT algorithm $\mathcal{A}$ given by:

$$1 \leftarrow \mathcal{A}(y_1 \cdots y_n y_{n+1}) \quad \Leftrightarrow \quad y_{n+1} = y_1 \oplus \cdots \oplus y_n$$

for all $n \geq 0$ and $y \in \{0,1\}^{n+1}$. Then,

One may calculate then the same probabilities as in the previous examples.

Pseudo-random generators with one bit expansion

RSA PRG

Example 9 (The RSA function)

1. RSA permutation: $RSA_{n,e} : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$

$$RSA_{n,e}(x) = x^e \bmod n,$$

where $n = pq$ is an RSA modulus, $e \in \mathbb{Z}_{\phi(n)}^*$, and $x \in \mathbb{Z}_n^*$;

2. RSA assumption: Given (n, e, y), no PPT adversary can compute x with $x^e \equiv_n y$ with non-negligible probability;

3. RSA PRG: Under the RSA assumption,

$$G(x) = RSA_{n.e}(x) \parallel LSB(x)$$

is a PRG with one bit expansion.

Rabin PRG

Example 10 (The Rabin function)

1. Rabin function: $R_n : \mathbb{Z}_n^* \to QR_n$

$$R_n(x) = x^2 \bmod n,$$

where $n = pq$ is an RSA modulus and $x \in \mathbb{Z}_n^*$.

Restricted to quadratic residues modulo Blum moduli n, R_n is a permutation, called Rabin permutation;

2. Factoring assumption: No PPT adversary can factorize sufficiently large *RSA* moduli with non-negligible probability;

3. Rabin PRG : Under the factoring assumption,

$$G(x) = R_n(x) \parallel LSB(x)$$

is a PRG with one bit expansion, if R_n is the Rabin permutation.

DL PRG

Example 11 (The Discrete Logarithm function)

1. DL function: $DL_{p,g} : \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$

$$DL_{p,g}(x) = g^x \bmod p,$$

where p is a prime and g is a generator of $\mathbb{Z}_p^*$;

2. DL assumption: No PPT adversary can solve DLP in $\mathbb{Z}_p^*$ with non-negligible probability for sufficiently large p;
3. DL PRG : Under the DL assumption,

$$G(x) = DL_{p,g}(x) \parallel half_{p-1}(x)$$

is a PRG with one bit expansion if $DL_{p,g}$ is extended to a permutation, where

$$half_{p-1}(x) = \begin{cases} 1, & \text{if } 0 \leq x \leq (p-1)/2 \\ 0, & \text{otherwise.} \end{cases}$$

Pseudo-random generators with polynomial expansion

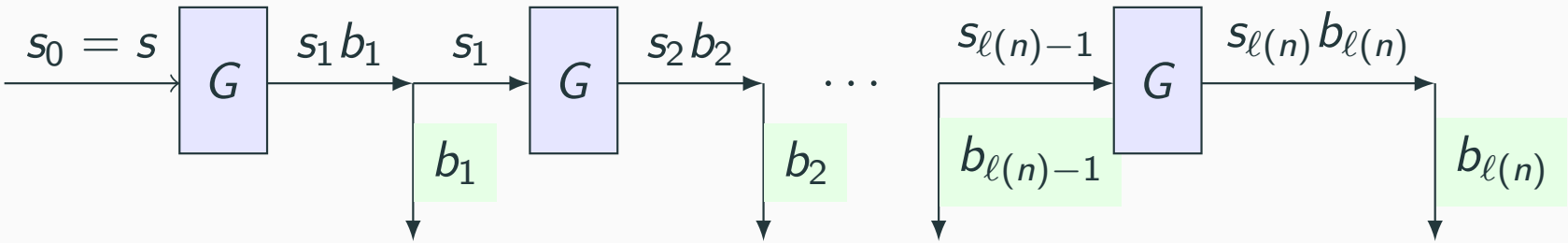
PRG with polynomial expansion factor

Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a function and ℓ be a strictly increasing positive polynomial. Define $G' : \{0,1\}^n \leftarrow \{0,1\}^{\ell(n)}$ by

$$G'(s) = b_1 \cdots b_{\ell(n)}$$

where

1. $s_0 = s$
2. $G(s_i) = s_{i+1} \parallel b_{i+1}$, for all $0 \leq i < \ell(n)$



Theorem 12

The function G' is a PRG with expansion factor $\ell(n)$, provided that G is a PRG (with one bit expansion).

The RSA PRG

Example 13 (RSA PRG)

1. Use the seed to generate an RSA modulus n, $e \in \mathbb{Z}^*_{\phi(n)}$, and $x \in \mathbb{Z}^*_n$;
2. Output

$$LSB(x) \parallel LSB(x^e \bmod n) \parallel LSB((x^e)^e \bmod n) \parallel \cdots$$

Under the RSA assumption, G is a PRG.

The Blum-Micali PRG

Example 14 (Blum-Micali PRG)

1. Use the seed to generate a prime $p = 2q + 1$, a generator g of $\mathbb{Z}_p^*$, and $x \in \mathbb{Z}_p^*$, where q is a prime too;
2. Output

$$half_{p-1}(x) \parallel half_{p-1}(g^x \bmod p) \parallel half_{p-1}(g^{g^x} \bmod p) \parallel \cdots$$

Under the DL assumption, G is a PRG.

The Blum-Blum-Schub PRG

Example 15 (Blum-Blum-Schub PRG)

1. Use the seed to generate a Blum modulus $n = pq$ and $x \in QR_n$;
2. Output

$$LSB(x) \parallel LSB(x^2 \bmod n) \parallel LSB((x^2)^2 \bmod n) \parallel \cdots$$

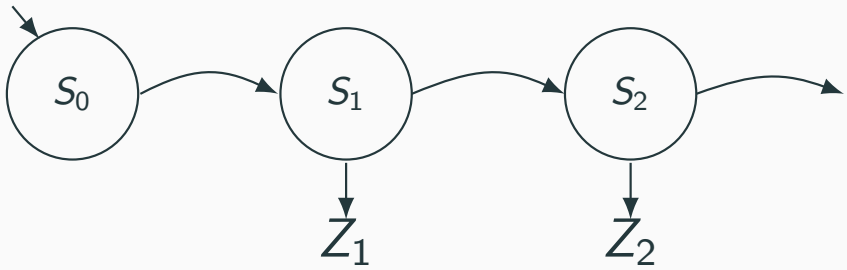
Under the factoring assumption, G is a PRG.

The BBS generator is the most efficient of the generators discussed so far. Even so, it is slow. A possibility to make it more efficient is to extract more bits at one iteration. For a discussion on this topic, we indicate [1, 7].

PRGs in practice

PRGs in practice

1. The generators above are slow because they use number theory techniques that involve multiplication or exponentiation operations;
2. In practice, fast generators are desired. They are based on light operations (modular addition, shifts, XOR). This usually leads to a loss of security that becomes evident over the years;
3. Most generators have an architecture of finite automaton with outputs



4. In practice, the security of a generator is certified by submitting them to a battery of statistical tests such as DieHard [6], TestU01 [5] or NIST statistical test battery [2].

PRGs in practice

1. Examples of generators proposed over time that have proven to be insecure:
   - The generator that is the basis of the RC4 stream cipher, used in multiple applications such as WEP, SSL/TLS and so on;
   - The generator that is the basis of the CSS DVD protection system;
   - The generators from A5/1, A5/2, and A5/3 GSM stream ciphers;
   - The generator from E0 stream cipher used in the Bluetooth protocol;
2. Examples of modern generators: the generators that form the basis of stream ciphers SALSA and CHACHA.

Some of these generators will be discussed during the seminars!

Reading and exercise guide

Reading and exercise guide

Course readings:

1. Pages 60-64 and 296-298 from [4].

For a comprehensive treatment of the field, I recommend Chapter 3 of [3].

References

[1] Alexi, W., Chor, B., Goldreich, O., and Schnorr, C. C. (1988). RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209.

[2] Bassham, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., Barker, E. B., Leigh, S. D., Levenson, M., Vangel, M., Banks, D. L., Heckert, N. A., Dray, J. F., and Vo, S. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Gaithersburg, MD, USA.

[3] Goldreich, O. (2004). *Foundations of Cryptography. Basic Tools*. Cambridge University Press, first edition.

[4] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

[5] L'Ecuyer, P. and Simard, R. (2007). TestU01: AC library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):1–40.

[6] Marsaglia, G. (1995). The Marsaglia random number CDROM: including the diehard battery of tests of randomness. Technical report, Florida State University.

[7] Sidorenko, A. and Schoenmakers, B. (2005). Concrete security of the Blum-Blum-Shub pseudorandom generator. In *Proceedings of the 10th International Conference on Cryptography and Coding*, IMA'05, page 355?375, Berlin, Heidelberg. Springer-Verlag.

Introduction to Symmetric Key Encryption

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Introduction

Computational security

1. Perfect security is impractical: keys must be as long as messages
2. We would like to securely encrypt arbitrary long messages using short keys
3. How ? Relax the security requirements:
   - 3.1 Consider only adversaries that use a reasonable amount of time and memory
   - 3.2 Consider a weaker notion of security such as concrete security or asymptotic security
4. Concrete security: a scheme is (t, ϵ)-secure if every adversary running for time at most t succeeds in breaking the scheme with probability at most ϵ
5. Asymptotic security: a scheme is secure if every PPT adversary succeeds in breaking the scheme with only negligible probability

Computational security

1. We follow the asymptotic security approach
2. Adversary: PPT algorithm
3. Security parameter: used to measure the input size of the computational problem
   - 3.1 Resource requirements by the cryptographic algorithm or protocol
   - 3.2 Running time of the adversary as well as its success probability

Symmetric key encryption

Setting up the scene

Goal:

1. Introduce the concept of a symmetric encryption scheme
2. Define security models
3. Discuss stream ciphers
4. Discuss block ciphers

Terminology and notation from previous lectures:

1. $\mathcal{K}$ denotes a set of keys
2. $\mathcal{M}$ denotes a set of (plain) messages, also called plaintexts
3. $\mathcal{C}$ denotes a set of ciphertexts

Symmetric key encryption

Definition 1

A symmetric key encryption (SKE) scheme over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ is a triple of algorithms $\mathcal{S} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ such that:

1. $\mathcal{G}(\lambda)$: PPT algorithm, called the key generation algorithm, which outputs a key $K \in \mathcal{K}$ when invoked on a security parameter λ;
2. $\mathcal{E}(K, m)$: PPT algorithm, called the encryption algorithm, which outputs a ciphertext $c \in \mathcal{C}$ when invoked on a key $K \in \mathcal{K}$ and a message $m \in \mathcal{M}$;
3. $\mathcal{D}(K, c)$: deterministic PT algorithm, called the decryption algorithm, which outputs a message $m \in \mathcal{M}$ or a special symbol $\perp$ (denoting failure) when invoked on a key $K \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$.

Soundness:

$$(\forall \lambda)(\forall m \in \mathcal{M})(\forall c \in \mathcal{C})(K \leftarrow \mathcal{G}(\lambda) \wedge c \leftarrow \mathcal{E}(K, m) \Leftarrow m := \mathcal{D}(K, c))$$

Symmetric key encryption

1. The complexity of $\mathcal{G}$ is measured w.r.t. the input λ
2. The complexity of $\mathcal{E}$ ($\mathcal{D}$) is measured w.r.t. $|K| + |m|$ ($|K| + |c|$). The key size is polynomial in λ, but $|m|$ ($|c|$) might not be
3. We will generally consider encryption/decryption of polynomial length messages in λ. As a result, all algorithms will run in time complexity polynomial in λ
4. When $\mathcal{E}$ is defined only for messages of length $\ell(\lambda)$, for some polynomial ℓ, we will say that the encryption scheme is a fixed-length encryption scheme with length parameter ℓ

Symmetric key encryption

Terminology, notation, remarks:

1. Symmetric key encryption: the encryption key is also the decryption key
2. Symmetric key encryption is also called private-key encryption. When two parties want to use an SKE scheme, it is assumed they are in the possession of the same key K
3. The encryption algorithm may be stateful: it maintains a state, initialized in some pre-defined way, and computes the ciphertext based on the key, message, and its current state. If no state is maintained, the algorithm is called stateless
4. Both randomized and stateful encryption algorithms are rare in practice

Security models

Security models

Definition 2

A security model is a pair consisting of a security goal and an attack model.

Standard security goals:

1. Semantic security (SS)
2. Indistinguishability (IND)
3. Non-malleability (NM)

Standard attack models:

1. Chosen plaintext attack (CPA)
2. Non-adaptive chosen ciphertext attack (CCA1)
3. Adaptive chosen ciphertext attack (CCA2)

Security goals

1. Semantic security was proposed by Goldwasser and Micali in 1982, and it was the first definition of security for encryption. It formalizes the fact that no adversary can obtain any partial information about the message of a given ciphertext. In other words, whatever can efficiently be computed about a message from its ciphertext can also be computed without the ciphertext
2. Semantic security is a "polynomially bounded" version of the concept of perfect secrecy introduced by Shannon in 1949
3. Semantic security is complex and difficult to work with
4. Indistinquishability is an equivalent definition to semantic security which is somewhat simpler
5. Non-malleability means that, given a ciphertext c of some message m, no efficient adversary can construct another ciphertext c' of some message m' meaningfully related to m

Attack models

Some common attack models are:

1. Passive attacks (eavesdropping)
2. Active attacks:
   - 2.1 Chosen plaintext attack (CPA): $\mathcal{A}$ has access to the encryption oracle (this is for free for PKE)
   - 2.2 Non-adaptive chosen ciphertext attack (CCA1): $\mathcal{A}$ has, in addition to the ability of a CPA adversary, access to a decryption oracle before the challenge phase
   - 2.3 Adaptive chosen ciphertext attack (CCA2): $\mathcal{A}$ has, in addition to the ability of a CCA1 adversary, access to a decryption oracle after the challenge phase. However, no decryption query is allowed involving the challenge ciphertext

Relationships among security models

By combining a security goal X with an attack model Y, we obtain a security model X-Y. A few remarks on these combinations are in order:

1. Indistinguishability in the presence of an eavesdropper is the weakest form of security where the adversary can only eavesdrop on ciphertexts

2. We will mainly focus on security models where the adversary is active and not only passive

Security models

The diagram below only aims to create an image on the relationships between the security models introduced so far (an arrow means an implication). Some of these relationships are far from trivial.

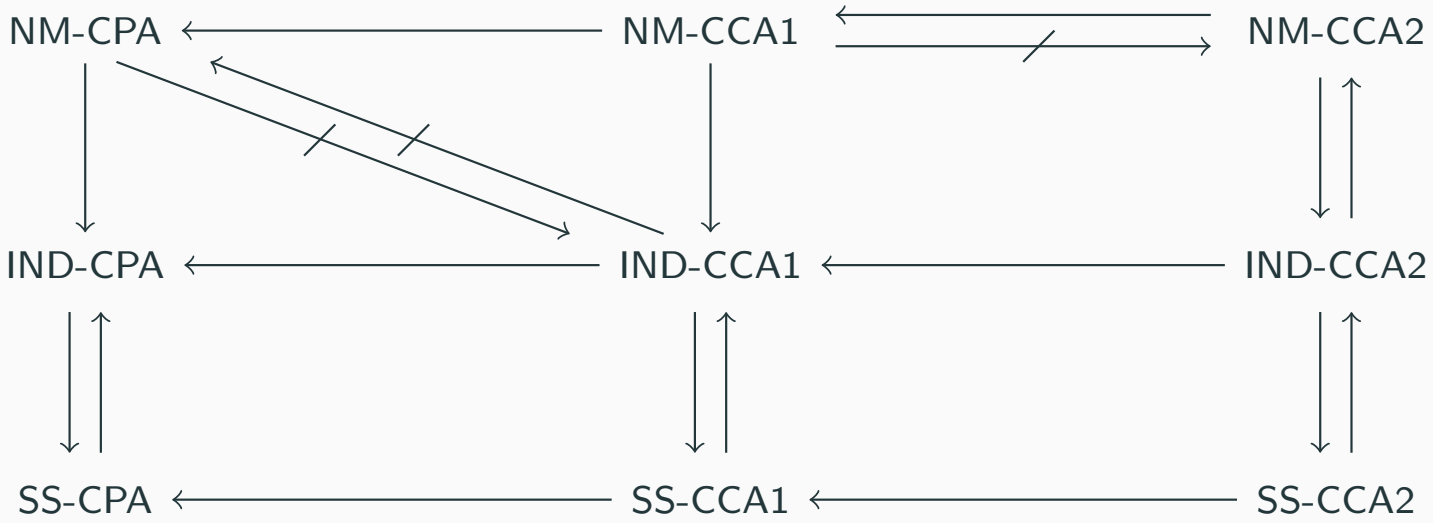


Figure 1: Relationships among security models

IND for single encryption

Experiment $PrivK^{ind\text{-}se\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda)$, where $b \in \{0,1\}$

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$

2: The adversary $\mathcal{A}$ submits a pair $(m_0, m_1) \in \mathcal{M}^2$ with $|m_0| = |m_1|$

3: The challenger computes $c \leftarrow \mathcal{E}(K, m_b)$ and sends it to $\mathcal{A}$

4: The adversary outputs a bit $b' \in \{0,1\}$

5: Return b'.

Remark 3

$P(PrivK^{ind\text{-}se\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda) = b')$ is the probability that $PrivK^{ind\text{-}se\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda)$ returns b', which is also the probability that $\mathcal{A}$ returns b' in this experiment.

The probability is taken over the internal coin tosses of all algorithms!

IND for multiple encryptions

Experiment $PrivK^{ind\text{-}me\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda)$, where $b \in \{0,1\}$

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$

2: The adversary $\mathcal{A}$ submits a sequence of pairs $(m_0^1, m_1^1), \ldots, (m_0^\ell, m_1^\ell) \in \mathcal{M}^2$ with $|m_0^i| = |m_1^i|$ for all i

3: The challenger computes $c^i \leftarrow \mathcal{E}(K, m_b^i)$ for all i and sends $c^1, \ldots, c^\ell$ to $\mathcal{A}$

4: The adversary outputs a bit $b' \in \{0,1\}$

5: Return b'.

Remark 4

$P(PrivK^{ind\text{-}me\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda) = b')$ is the probability that $PrivK^{ind\text{-}me\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda)$ returns b', which is also the probability that $\mathcal{A}$ returns b' in this experiment.

The probability is taken over the internal coin tosses of all algorithms!

IND-CPA security games

Experiment $PrivK^{ind\text{-}cpa\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda)$, where $b \in \{0,1\}$

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$

2: Repeat the following two steps a polynomial number of times

- $\mathcal{A}$ submits a pair $(m_0, m_1) \in \mathcal{M}^2$ with $|m_0| = |m_1|$
- The challenger computes $c \leftarrow \mathcal{E}(K, m_b)$ and sends it to $\mathcal{A}$

3: The adversary outputs a bit $b' \in \{0,1\}$

4: Return b'.

Remark 5

$P(PrivK^{ind\text{-}cpa\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda) = b')$ is the probability that $PrivK^{ind\text{-}cpa\text{-}b}_{\mathcal{A},\mathcal{S}}(\lambda)$ returns b', which is also the probability that $\mathcal{A}$ returns b' in this experiment.

The probability is taken over the internal coin tosses of all algorithms!

IND security

Definition 6

An SKE scheme $\mathcal{S}$ is IND/IND-CPA secure if

$$\left|P(PrivK^{ind\text{-}x\text{-}0}_{\mathcal{A},\mathcal{S}}(\lambda)=1)-P(PrivK^{ind\text{-}x\text{-}1}_{\mathcal{A},\mathcal{S}}(\lambda)=1)\right|$$

is negligible, for all PPT algorithms $\mathcal{A}$, where $x \in \{se, me, cpa\}$.

IND security: bit guessing version

Experiment $PrivK^{ind\text{-}cpa}_{\mathcal{A},\mathcal{S}}(\lambda)$ % XXX=CPA

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$

2: Access to the encryption oracle : repeat the following two steps a polynomial number of times
   - $\mathcal{A}$ submits a query $m \in \mathcal{M}$ to the challenger
   - The challenger computes $c \leftarrow \mathcal{E}(K, m)$ and sends it to $\mathcal{A}$

3: Challenge :
   - $\mathcal{A}$ sends a pair $(m_0, m_1) \in \mathcal{M}^2$ with $|m_0| = |m_1|$
   - The challenger generates a bit $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathcal{E}(K, m_b)$, and sends c to $\mathcal{A}$

4: Access to the encryption oracle : $\mathcal{A}$ has access to the encryption oracle (as above) for messages different than m_0 and m_1

5: Guess : The adversary outputs a bit $b' \in \{0,1\}$

6: If $b' = b$ then return 1 else return 0

IND security: bit guessing version

Consider the CPA case (similar for the other cases)

Definition 7

The IND-CPA advantage of $\mathcal{A}$ with respect to the SKE scheme $\mathcal{S}$ is

$$Adv^{ind\text{-}cpa}_{\mathcal{A},\mathcal{S}}(\lambda) = \left| P(PrivK^{ind\text{-}cpa}_{\mathcal{A},\mathcal{S}}(\lambda) = 1) - \frac{1}{2} \right|$$

Proposition 8

For any SKE scheme $\mathcal{S}$ and any PPT adversary $\mathcal{A}$, the following property holds:

$$Adv^{ind\text{-}cpa}_{\mathcal{A},\mathcal{S}}(\lambda) =$$

$$\frac{1}{2} \left| P(PrivK^{ind\text{-}cpa\text{-}0}_{\mathcal{A},\mathcal{S}}(\lambda) = 1) - P(PrivK^{ind\text{-}cpa\text{-}1}_{\mathcal{A},\mathcal{S}}(\lambda) = 1) \right|$$

Symmetric Key Cryptography

Stream Ciphers

Prof.dr. Ferucio Laurenţiu Ţiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iaşi
Iaşi 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Introduction

Stream ciphers

Main characteristics of a stream cipher:

- A stream cipher encrypts streams of plaintext characters
- A character in this case is an element of an alphabet of a very limited size that can efficiently be enumerated in practice (e.g., $\{0,1\}$ or $\{0,1\}^8$)
- In a stream cipher, a message m is encrypted by a key K of the same length ($|K| = |m|$), usually called a keystream
- The encryption is character-driven
- OTP is a stream cipher, but quite impractical
- Practical stream ciphers must have a keystream generator initially seeded with a short secret key

Stream ciphers

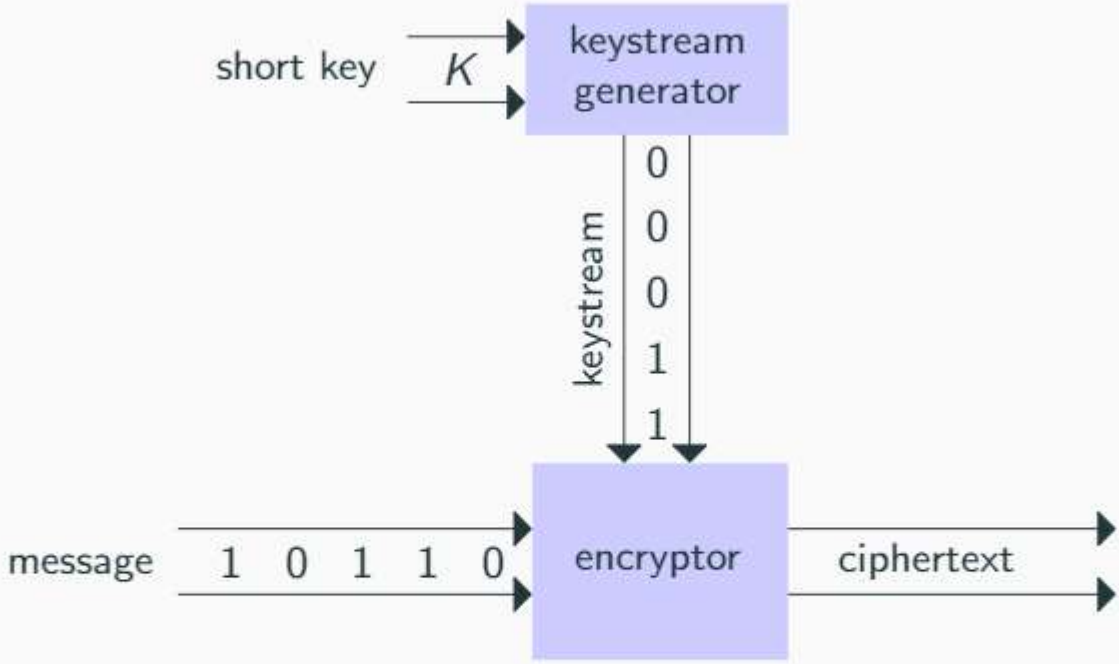


Figure 1: Pictorial view of a stream cipher

Stream ciphers from PRG

PRG as keystream generators

Cipher 1 (Stream cipher from PRG)

Let G be a PRG with expansion factor ℓ. Define an SKE scheme $\mathcal{S}(G) = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ over $(\{0,1\}^\lambda, \{0,1\}^{\ell(\lambda)}, \{0,1\}^{\ell(\lambda)})$ by:

1. $\mathcal{G}(\lambda)$: *output* K, *where* $K \leftarrow \{0,1\}^\lambda$
2. $\mathcal{E}(K, m)$: *output* $c = m \oplus G(K)$
3. $\mathcal{D}(K, c)$: *output* $m = c \oplus G(K)$

Theorem 1

The SKE scheme in Description 1 is IND secure for single encryption (provided that G is a PRG).

Stream ciphers: using the same key twice

Using the same key twice (two-time pad):

- If $c_1 = m_1 \oplus G(K)$ and $c_2 = m_2 \oplus G(K)$, then $c_1 \oplus c_2 = m_1 \oplus m_2$
- Natural language text contains enough redundancy to allow the adversary to recover m_1 and m_2 from $c_1 \oplus c_2$

  [6] describes a method that uses a statistical language model and a dynamic programming algorithm. The results are quite impressive:

  " ... if only the type of each message is known (e.g. an HTML page in English) ... (our method) produces up to 99% accuracy on realistic data and can process ciphertexts at 200ms per byte on a $2,000 PC."

Stream ciphers: using the same key twice

Real scenario: Venona Project

- Initiated on February 1, 1943, by Gene Grabeel (American mathematician and cryptanalyst)
- Goal: analyze encrypted Soviet diplomatic messages
- Encryption:
  - Use a code book to translate letters, words, and phrases into numbers
  - Use then a one-time pad
- Pitfall of use: some of the one-time pad material had incorrectly been reused by the Soviets (entire pages), which allowed partial decryption of the traffic

Stream ciphers: using the same key twice

Real scenarios:

- Microsoft implementation of PPTP in Windows NT uses RC4. Its original implementation uses the same key to encrypt messages from A to B and from B to A (see [7])
- Microsoft have used RC4 to protect Word and Excel document. When encrypted documents were modified and saved, the same key was used (see [10])

Never use the same key to encrypt more than one message with stream ciphers!

Stream ciphers: malleability

Malleability:

- From an encryption $c = m \oplus G(K)$ of m one can simply obtain an encryption of $m \oplus m'$ by $c' = c \oplus m'$

Real scenarios:

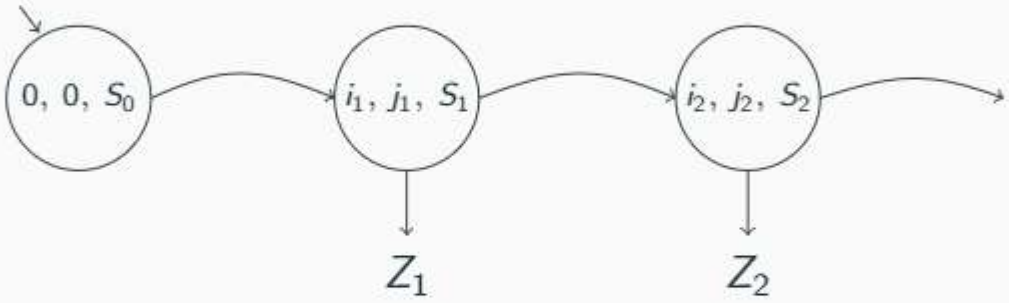
- Assume that the adversary knows a prefix m_1 of m (m_1 might be a standard header filled with someone's address, name, etc.)
- The adversary wants to replace m_1 by m_2 (m_2 might be a header filled with information up to his desire)
- The adversary may compute $c \oplus (m_1 \oplus m_2)0 \cdots 0$ to obtain what he wants

Stream ciphers do not guarantee integrity!

Practical stream ciphers

The stream cipher RC4

1. Designed by Ronald Rivest in 1987
2. Kept as a commercial secret until 1994 (when it was disclosed)
3. Works as a finite state automaton with outputs



4. A state is a triple (i, j, S) consisting of two bytes i and j and a permutation $S = S[0] \cdots S[255]$ of the set $\{0,1\}^8$ of all bytes
5. The initial permutation S_0 is prepared by the algorithm `Init` that depends on an ℓ-byte key $K = K[0] \cdots K[\ell - 1]$, where $5 \leq \ell \leq 16$
6. The transition relation and the output function is given by the algorithm `Trans`

RC4: preparing the initial state

Init

Input: a key K of ℓ bytes

Output: initial state $(0, 0, S_0)$, where S_0 is a permutation of $\{0,1\}^8$

begin

1: $j := 0$

2: $S_0 := id$ ▷ id is the identity permutation

3: **for** $i := 0$ to 255 **do**

4: $\quad j := (j + S_0[i] + K[i \bmod \ell]) \bmod 256$

5: $\quad$ swap $S_0[i]$ and $S_0[j]$

6: **end for**

7: return $(0, 0, S_0)$

end

RC4: the transition function

Trans

Input: state (i, j, S)

Output: new state and an output byte

begin

1: $i := i + 1$

2: $j := j + S[i]$

3: swap $S[i]$ and $S[j]$

4: return new state (i, j, S) ▷ In fact, this is kept internally

5: return byte $S[S[i] + S[j] \bmod 256]$

end

Encryption and decryption by RC4

1. Denote by $RC4\_gen$ the RC4 keystream generator
   - 1.1 when seeded by some key K, $RC4\_gen(K)$ prepares its initial state (and maintains it internally);
   - 1.2 when invoked, $RC4\_gen(K)$ generates a new state and outputs a byte (and maintains the current state internally)
2. RC4 encryption: on an n-byte message $m = m_1 \cdots m_n$, the generator $RC4\_gen(K)$ outputs an n-byte keystream $Z_1 \cdots Z_n$. The ciphertext is $c = (m_1 \oplus Z_1) \cdots (m_n \oplus Z_n)$
3. RC4 decryption: on an n-byte ciphertext $c = c_1 \cdots c_n$, the generator $RC4\_gen(K)$ outputs an n-byte keystream $Z_1 \cdots Z_n$. The message is $m = (c_1 \oplus Z_1) \cdots (c_n \oplus Z_n)$

RC4 in practice

1. RC4 is suited for software implementations
2. RC4 was used in a large variety of applications: SSL/TLS, WEP, WPA, MS-PPTP etc.
3. Recent results have shown that the *RC4_gen* output is biased:
   - 3.1 $P(Z_2 = 0x00) \approx \frac{1}{128}$ (see [3])
   - 3.2 $P(Z_r = 0x00) \approx \frac{1}{256} + \frac{c_r}{256^2}$ for $3 \leq r \leq 255$, where $c_3 = 0.351089$ and $0.242811 \leq c_r \leq 1.337057$ for $r \geq 4$ (see [8])
   - 3.3 See also [1]
4. Several other variants of RC4 have been proposed: *RC4A*, *VMPC*, *RC4*$^+$, *Spritz*

Practical stream ciphers: LFSR-based constructions

Many practical stream ciphers have been built by using LFSRs as keystream generators. A Linear Feedback Shift Register (LFSR) of length n can be viewed as a finite automaton with output as follows:

1. each state is an n-bit stream (vector) usually called register
2. the transition function δ is given by

$$\delta(r_0, r_1, \ldots, r_{n-1}) = (r_1, \ldots, r_{n-1}, f(r_0, \ldots, r_{n-1}))$$

   where $f(x_0, \ldots, x_{n-1}) = c_0 x_0 \oplus \cdots \oplus c_{n-1} x_{n-1}$ is a Boolean function ($c_0, \ldots, c_{n-1}$ are Boolean constants with $c_0 \neq 0$).

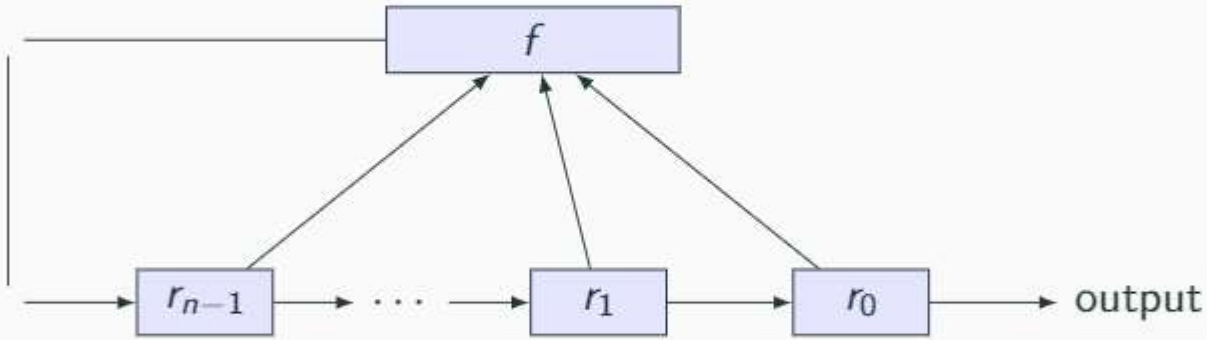
   f is called the feedback function, each position i with $c_i \neq 0$ is called a tap position, and a transition step is called a cycle

3. the output function g is given by

$$g(r_0, \ldots, r_{n-1}) = r_0$$

Practical stream ciphers: LFSR-based constructions

The diagram below shows how LFSR works. When the register is clocked (a transition step is applied), a new bit value is computed by f, the leftmost bit is outputted, and the new bit is inserted to the right by shifting all bits a position to the left.



Practical stream ciphers: CSS

We present here CSS just to illustrate how LFSR are used to design a keystream generator:

1. CSS = Content Scrambling System
2. It was designed in 1980's for preventing unauthorized duplication of DVDs
3. It is based on two LFSRs: one is of length 17 with the feedback function $f_1(x_0, \ldots, x_{16}) = x_0 \oplus x_{14}$, and the other one is of length 25 with the feedback function $f_2(x_0, \ldots, x_{24}) = x_0 \oplus x_3 \oplus x_4 \oplus x_{12}$
4. The keystream generator runs in parallel the two LFSR and combines their outputs by addition modulo 256 (the `Trans` algorithm)
5. The initial state of the keystream generator is prepared from a given 40-bit secret key (the `Init` algorithm)

CSS: preparing the initial state

Init

Input: a key K of 40 bits

Output: $S_0 = (S_0^1, S_0^2, c) \in \{0,1\}^{17} \times \{0,1\}^{25} \times \{0,1\}$

begin

1: write $K = K_1 \parallel K_2 \in \{0,1\}^{16} \times \{0,1\}^{24}$

2: insert 1 on the 9th position in K_1 and on the 22nd position in K_2

3: initialize $LFSR_1$ by K_1

4: run $LFSR_1$ for 8 cycles and discard the output

5: $S_0^1 \leftarrow$ current state of $LFSR_1$

6: initialize $LFSR_2$ by K_2

7: run $LFSR_2$ by 16 cycles and discard the output

8: $S_0^2 \leftarrow$ current state of $LFSR_2$

9: $c \leftarrow 0$ ▷ carry bit

10: return $S_0 = (S_0^1, S_0^2, c)$

end

CSS: the transition function

Trans

Input: state (S_1, S_2, c)

Output: new state and an output byte

begin

1: run in parallel $LFSR_1$ and $LFSR_2$ from their states S_1 and S_2, resp., for 8 cycles and collect their outputs $x, y \in \{0,1\}^8$, resp.

2: $S_1 \leftarrow$ current state of $LFSR_1$

3: $S_2 \leftarrow$ current state of $LFSR_2$

4: return byte $x + y + c \bmod 256$

5: if $x + y > 255$ then $c \leftarrow 1$ else $c \leftarrow 0$ ▷ carry bit

6: return new state (S_1, S_2, c) ▷ kept internally

end

CSS in practice

1. The encryption by CSS works by XOR-ing the plaintext with the keystream;
2. CSS can be brute-force attacked in time 2^{40} (the seed space size);
3. Faster attack to recover the seed: time 2^{16} [9].

Other practical stream ciphers

1. A5/1, A5/2, A5/3 stream ciphers for GSM encryption
   - 1.1 They are based on three LFSRs
   - 1.2 All have been cryptanalysed by several researchers (see [2])
2. E0 stream cipher for Bluetooth encryption
   - 2.1 It is based on four LFSRs of length 25, 31, 33, and 39 bits (total length $= 128$ bits)
   - 2.2 The most efficient cryptanalysis requires the first 24 bits of $2^{23.8}$ frames (a frame is 2745 bits long) and 2^{38} computations to recover the key (see [5])
3. Salsa, designed by Bernstein in 2005
4. ChaCha, designed by Bernstein in 2008

Reading and exercise guide

Reading and exercise guide

Course readings:

1. Pages 43-69 from [4].

I recommend reading the articles mentioned in the bibliography to create the best possible image of the field.

References

[1] AlFardan, N., Bernstein, D. J., Paterson, K. G., Poettering, B., and Schuldt, J. C. N. (2013). On the security of RC4 in TLS. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 305–320, Washington, D.C. USENIX Association.

[2] Barkan, E., Biham, E., and Keller, N. (2003). Instant ciphertext-only cryptanalysis of GSM encrypted communication. In Boneh, D., editor, *Advances in Cryptology - CRYPTO 2003*, pages 600–616, Berlin, Heidelberg. Springer Berlin Heidelberg.

[3] Fluhrer, S. R., Mantin, I., and Shamir, A. (2001). Weaknesses in the key scheduling algorithm of RC4. In *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, SAC '01, pages 1–24, Berlin, Heidelberg. Springer-Verlag.

[4] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

[5] Lu, Y., Meier, W., and Vaudenay, S. (2005). The conditional correlation attack: A practical attack on bluetooth encryption. In Shoup, V., editor, *Advances in Cryptology – CRYPTO 2005*, pages 97–117, Berlin, Heidelberg. Springer Berlin Heidelberg.

[6] Mason, J., Watkins, K., Eisner, J., and Stubblefield, A. (2006). A natural language approach to automated cryptanalysis of two-time pads. In *Proceedings of the 13th ACM Conference on Communication Security*, pages 235–244, New York, NY, USA. Association for Computing Machinery.

[7] Schneier, B. and Mudge (1998). Cryptanalysis of microsoft's point-to-point tunneling protocol (PPTP). In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, CCS 1998, page 132?141, New York, NY, USA. Association for Computing Machinery.

[8] Sen Gupta, S., Maitra, S., Paul, G., and Sarkar, S. (2014). (non-)random sequences from (non-)random permutations – analysis of RC4 stream cipher. *J. Cryptol.*, 27(1):67?108.

[9] Stevenson, F. A. (1999). Cryptanalysis of contents scrambling system. available at http://www.derfrosch.de/.

[10] Wu, H. (2005). The misuse of RC4 in Microsoft Word and Excel. *IACR Cryptol. ePrint Arch.*, 2005:7.

Symmetric Key Cryptography

Block Ciphers

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Block ciphers

Block ciphers: general considerations

- A block cipher is an SKE scheme where the encryption algorithm is deterministic and $\mathcal{M} = \mathcal{C}$. The elements of $\mathcal{M}$ are called data blocks or simply blocks
- Usually, the blocks have the same fixed length and their set is large and cannot be efficiently enumerated in practice
- A block cipher should be seen more as a pseudo-random permutation of a large set than as an encryption scheme (although the existing practical constructions do not quite meet this property)
- Block ciphers should be considered as building blocks for encryption schemes (or other cryptographic primitives)
- Examples of very important and popular block ciphers include DES and AES

Block ciphers

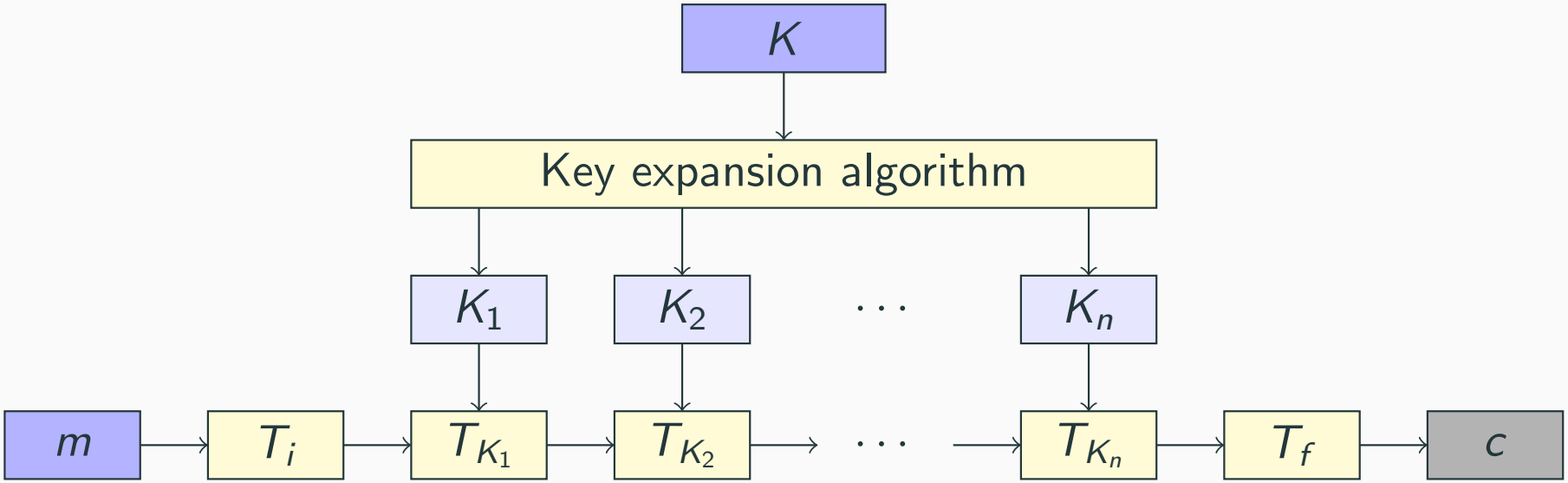


Figure 1: Pictorial view of a block cipher

Notation:

1. T_i = initial transformation
2. T_f = final transformation
3. T_{K_i} = transformation induced by K_i, $1 \leq i \leq n$

DES history

1. Data Encryption Standard (DES) is the best known and most widely used cryptosystem for civilian applications
2. May 15, 1973 : National Bureau of Standards solicited ciphers
3. March 17, 1975 : IBM developed a cipher by modifying an earlier cipher known as LUCIFER
4. January 15, 1977 : IBM proposal was accepted as a Data Encryption Standard (DES) for "unclassified" applications

DES mathematical description

1. $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^{64}$
2. for any key K,

$$e_K = P_0^{-1} \circ Rev \circ T_{K_{16}} \circ \cdots \circ T_{K_1} \circ P_0$$

and

$$d_K = P_0^{-1} \circ Rev \circ T_{K_1} \circ \cdots \circ T_{K_{16}} \circ P_0,$$

where:

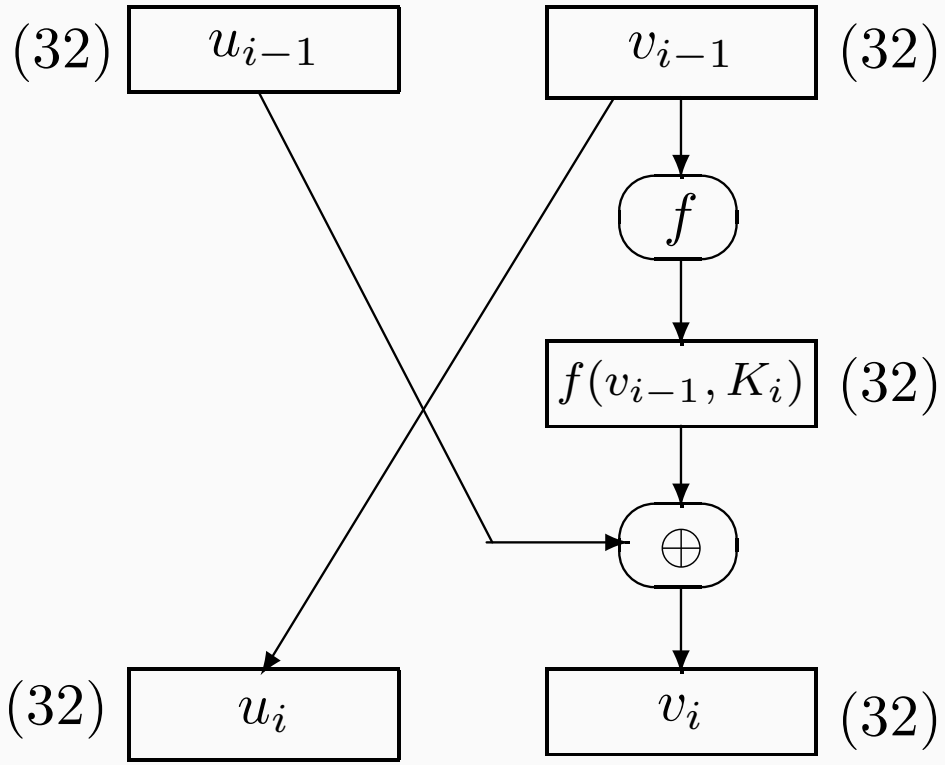
2.1 $P_0 : \{0,1\}^{64} \to \{0,1\}^{64}$ is a given initial permutation

2.2 $K_i \in \{0,1\}^{48}$ is computed from K, for all i (details later)

2.3 $T_{K_i}(uv) = v(u \oplus f(v, K_i))$, for any $u, v \in \{0,1\}^{32}$ (details later)

2.4 $Rev(uv) = vu$, for all $u, v \in \{0,1\}^{32}$

The transformation T_{K_i}



The function f

$f : \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$ is given by

$$f(v, X) = P(S(E(v) \oplus X))$$

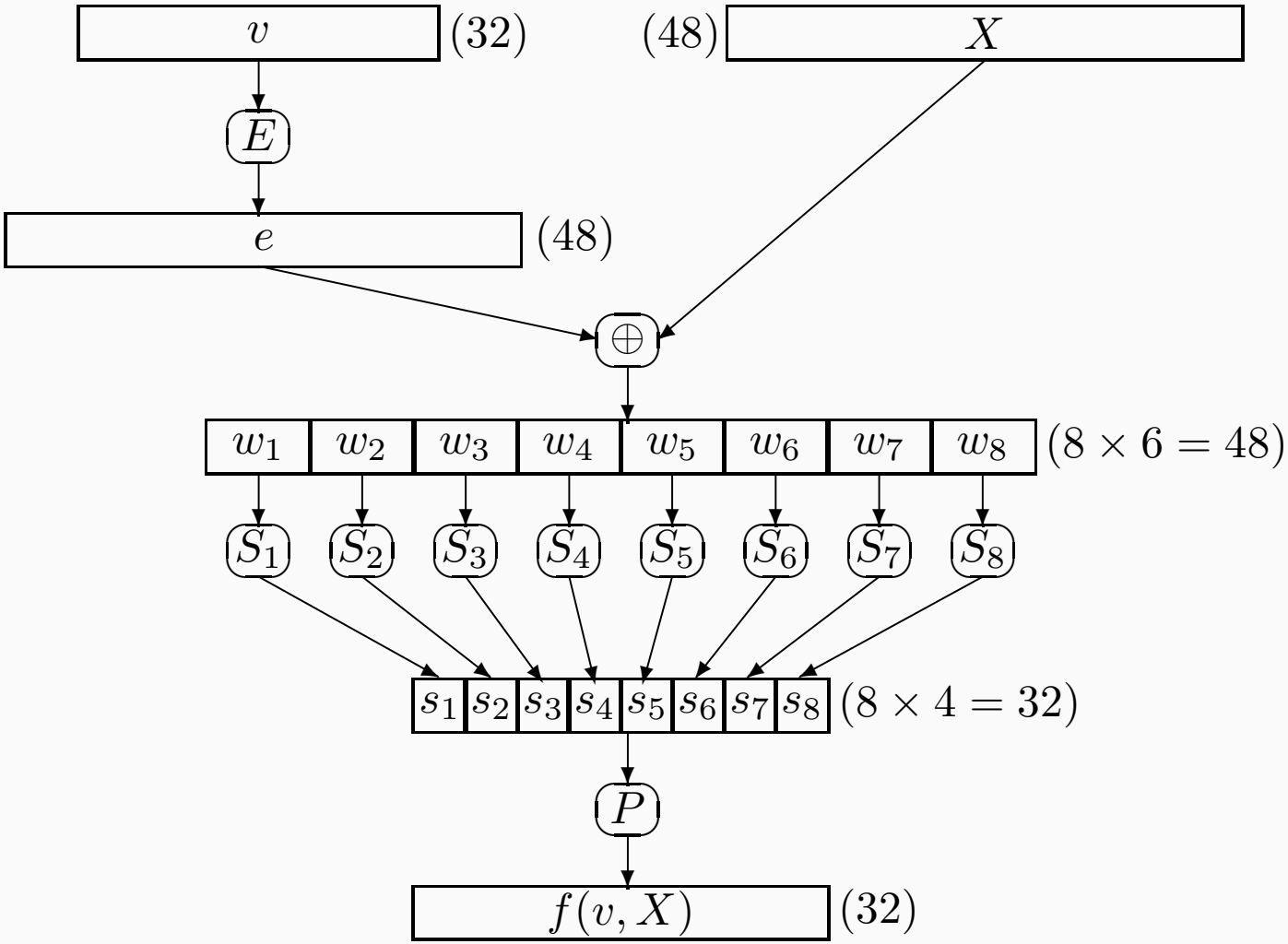
where:

1. $E : \{0,1\}^{32} \to \{0,1\}^{48}$ is a given function
2. $S : \{0,1\}^{48} \to \{0,1\}^{32}$ is a function given by

$$S(w) = S_1(w_1) \cdots S_8(w_8),$$

   where $w = w_1 \cdots w_8$ and $S_i : \{0,1\}^6 \to \{0,1\}^4$ are given for all i
3. $P : \{0,1\}^{32} \to \{0,1\}^{32}$ is a given permutation

The function f



Key scheduling

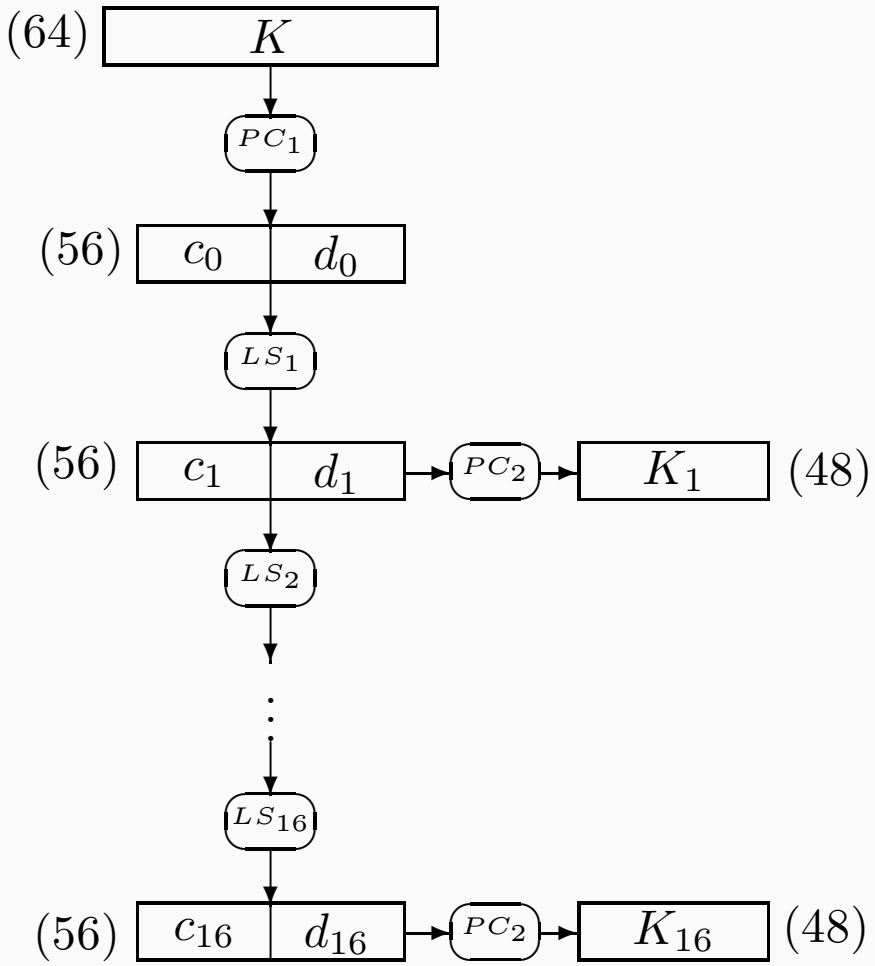
K_i, $1 \leq i \leq 16$, is a bitstring of length 48 computed as a function of the key K by

$$K_i = (PC_2 \circ LS_i \circ \cdots \circ LS_1 \circ PC_1)(K),$$

where:

1. $PC_1 : \{0,1\}^{64} \to \{0,1\}^{56}$ is a given function
2. $PC_2 : \{0,1\}^{56} \to \{0,1\}^{48}$ is a given function
3. $LS_j : \{0,1\}^{56} \to \{0,1\}^{56}$, for all $1 \leq j \leq i$, are given functions

Key scheduling



DES correctness

Proposition 1

1. $Rev \circ Rev = Id$.
2. $T_X \circ Rev \circ T_X = Rev$, *for any* $X \in \{0,1\}^{48}$.
3. $d_K(e_K(x)) = x$, *for any* $K, x \in \{0,1\}^{64}$.

Proof.

(1) follows directly from the definition of Rev, and (3) from (2) and (1).

(2) For any $u, v \in \{0,1\}^{32}$, the following equalities hold:

$$\begin{aligned} T_X(Rev(T_X(uv))) &= T_X((u \oplus f(v,X))v) \\ &= v(u \oplus f(v,X) \oplus f(v,X)) \\ &= vu \\ &= Rev(uv) \end{aligned}$$

Therefore, the equality $T_X \circ Rev \circ T_X = Rev$ is proved. □

DES security

1. DES key space is too small!
2. 1977: Whitefield Diffie and Martin Hellman suggested that a special-purpose computer can be build, capable of breaking DES by exhaustive search in a reasonable amount of time
3. 1980: Martin Hellman showed a time-memory trade-off that allows improvement over exhaustive search if memory space is plentiful
4. July 1998: Electronic Frontier Foundation (EFF) built a custom-designed chip and a personal computer, called *DES Cracker*, which was able to broke a DES-encoded message in 56 hours (budget of the project : $250,000)
5. Jan 1999: key-search improved to 22 hours through a combination of 100,000 networked PC's and the EFF machine

There was nothing terribly novel about DES Cracker except that it was built and DES's insecurity was definitely demonstrated!

DES properties

Proposition 2

1. *DES has the complementation property: if* $e_K(x) = y$ *then* $e_{\bar{K}}(\bar{x}) = \bar{y}$.
2. *DES has four weak keys* K *for which* $e_K \circ e_K = Id$.
3. *DES has six pairs of semi-weak keys* (K_1, K_2) *for which* $e_{K_1} \circ e_{K_2} = Id$.

Theorem 3 (Campbell and Wiener, 1992)

The set of DES permutations $G_{DES} = \{e_K | K \in \mathcal{K}\} \cup \{d_K | K \in \mathcal{K}\}$ *is not closed under functional composition. Moreover, a lower bound on the size of the group generated by* G_{DES} *is* 1.94×10^{2499}.

Strengthening DES against exhaustive search

1. 3DES in EDE mode
   - 1.1 $e_{K_1,K_2,K_3}(m) = e_{K_1}(d_{K_2}(e_{K_3}(m)))$
   - 1.2 Key space size $= 2^{168}$
   - 1.3 Three times slower than DES
2. DESX
   - 2.1 $e_{K_1,K_2,K_3}(m) = K_1 \oplus e_{K_2}(K_3 \oplus m)$
   - 2.2 Key space size $= 2^{184}$
3. DESX+ : replace $\oplus$ in DESX by addition mod 2^{64}

Advanced Encryption Standard (AES)

- Jan 2, 1997 : the American National Institute for Standards and Technology (NIST) solicited candidates for a new standard for the protection of sensitive electronic information
- Twenty-one teams of cryptographers from 11 countries submitted candidates
- Oct 2, 2000 : the winner was Rijndael designed by Joan Daemen and Vincent Rijmen
- Nov 26, 2001 : Rijndael was officially published as the Advanced Encryption Standard (AES)

The strong points of AES are:

- a simple and elegant design
- efficient and fast on modern processors, but also compact in hardware and on smartcards

AES description

- $\mathcal{M} = \mathcal{C} = \mathcal{M}_{4\times m}(\mathbb{Z}_2^8)$, where $m \in \{4, 6, 8\}$
- $\mathcal{K} = \mathcal{M}_{4\times k}(\mathbb{Z}_2^8)$, where $k \in \{4, 6, 8\}$
- For any $K \in \mathcal{K}$,

$$e_K = T^f_{K_n} \circ T_{K_{n-1}} \circ \cdots \circ T_{K_1} \circ T^i_{K_0}$$

and

$$d_K = T^{-f}_{K_0} \circ T^{-1}_{K_1} \circ \cdots \circ T^{-1}_{K_{n-1}} \circ T^i_{K_n}$$

- n denotes the number of rounds to be performed during the execution of the algorithm. It is dependent on the key and (message) block length

n	$m = 4$	$m = 6$	$m = 8$
$k = 4$	10	12	14
$k = 6$	12	12	14
$k = 8$	14	14	14

The transformations T_Z

T_Z^i, T_Z, and T_Z^f are given by:

- $T_Z^i = A_Z$
- $T_Z = A_Z \circ Mc \circ Sh \circ S$
- $T_Z^f = A_Z \circ Sh \circ S$

T_Z^{-1} and T_Z^{-f} are given by:

- $T_Z^{-1} = A_{Mc^{-1}(Z)} \circ Mc^{-1} \circ Sh^{-1} \circ S^{-1}$
- $T_Z^{-f} = A_Z \circ Sh^{-1} \circ S^{-1}$

for any $Z \in \mathcal{M}_{4\times m}(\mathbb{Z}_2^8)$.

The transformation A_Z

A_Z, called the AddRoundKey transformation, is just a simple bitwise XOR operation extended to matrices. That is,

$$A_Z(X)(i,j) = X(i,j) \oplus Z(i,j),$$

for any $X \in \mathcal{M}_{4\times m}(\mathbb{Z}_2^8)$, $0 \leq i \leq 3$ and $0 \leq j \leq m-1$

We simply write $A_Z(X) = X \oplus Z$

The transformation S

S, called the SubBytes transformation, is a non-linear byte substitution that operates independently on each byte of the input matrix. It uses a substitution table that can be computed by

$$S(X)(i,j)^t = M_1 \cdot X(i,j)' \oplus C,$$

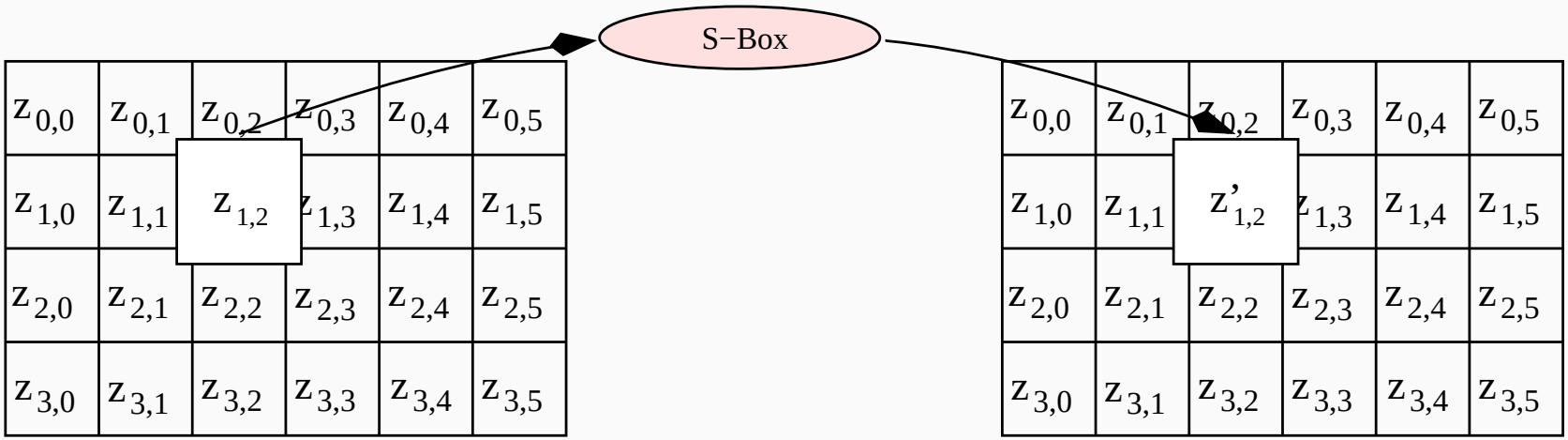
where:

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

The transformation S

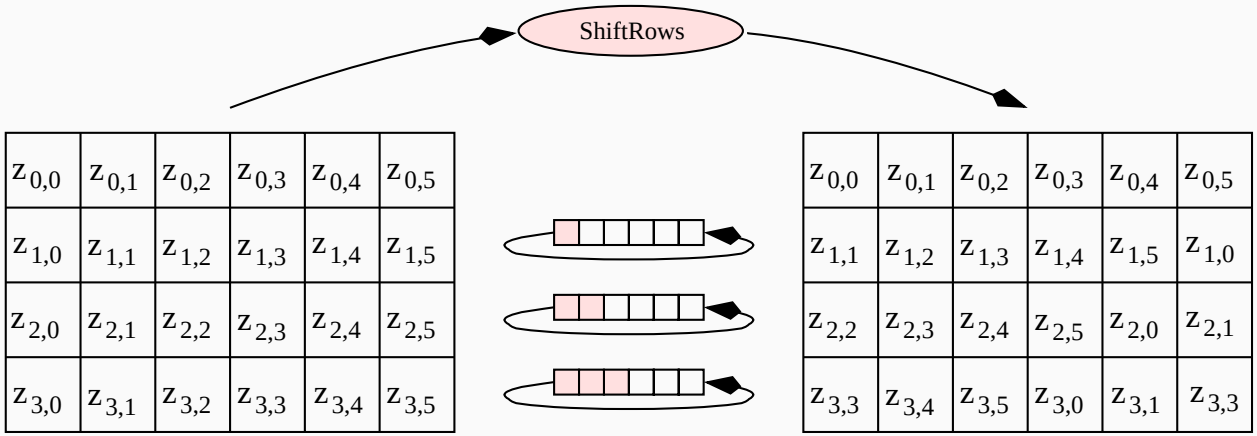
$$X(i,j)' = \begin{cases} (0,0,0,0,0,0,0,0)^t, & \text{if } X(i,j) = (00)_h \\ (X(i,j)^{-1})^t, & \text{otherwise} \end{cases}$$

(the inverse is in the finite field $GF(2^8)$ with the irreducible polynomial $x^8 + x^4 + x^3 + x + 1 \in \mathbb{Z}_2[x]$).



The transformation *Sh*

Sh, called the ShiftRows transformation, cyclically shifts the rows of the input matrix over different numbers of positions (offsets). The ith row is shifted over $C_i = i$ positions (i=0,1,2,3)



Formally, we may write

$$Sh(X)(i,j) = X(i,(j + C_i) \ mod \ m),$$

for any $X \in \mathcal{M}_{4\times m}(\mathbb{Z}_2^8)$, $0 \leq i \leq 3$ and $0 \leq j \leq m-1$;

The transformation *Mc*

Mc, called the MixColumns transformation, treats each column as a polynomial over $GF(2^8)$ and multiplies it modulo $x^4 + 1$ with a fixed polynomial $a(x)$ given by:

$$a(x) = (03)_h x^3 + (01)_h x^2 + (01)_h x + (02)_h.$$

This transformation can be written as a matrix multiplication in $GF(2^8)[x]$,

$$Mc(X) = M_2 \bullet X,$$

where

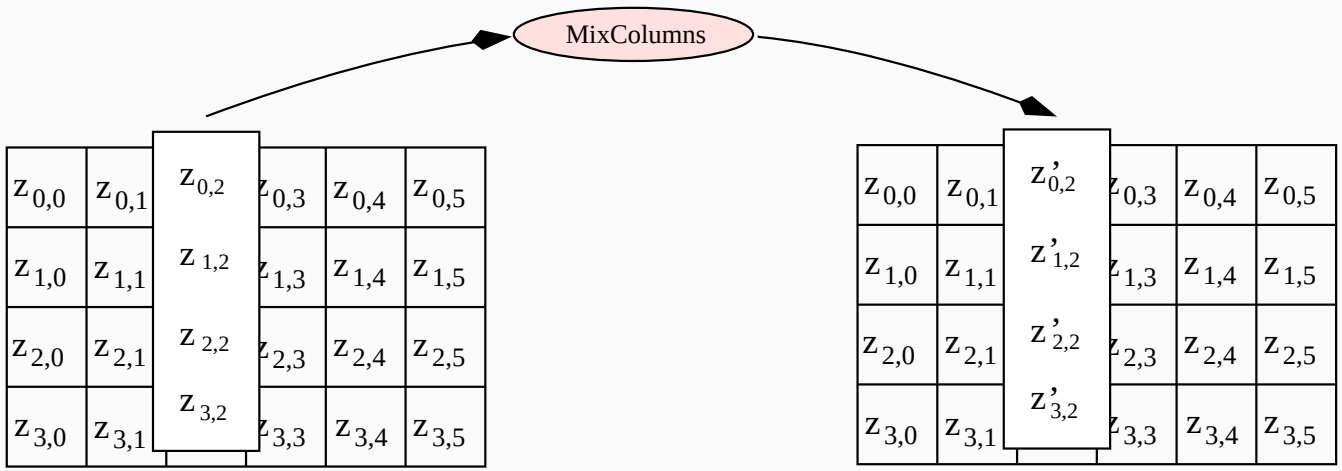
$$M_2 = \begin{pmatrix} (02)_h & (03)_h & (01)_h & (01)_h \\ (01)_h & (02)_h & (03)_h & (01)_h \\ (01)_h & (01)_h & (02)_h & (03)_h \\ (03)_h & (01)_h & (01)_h & (02)_h \end{pmatrix}$$

The transformation *Mc*

The matrix M_2 is invertible and its inverse is

$$M_2^{-1} = \begin{pmatrix} (0e)_h & (0b)_h & (0d)_h & (09)_h \\ (09)_h & (0e)_h & (0b)_h & (0d)_h \\ (0d)_h & (09)_h & (0e)_h & (0b)_h \\ (0b)_h & (0d)_h & (09)_h & (0e)_h \end{pmatrix}$$

The transformation is pictorially represented by



Round keys

$K_0, \ldots, K_n$, called the round keys, are obtained as follows:

- define first $W_0, W_1, \ldots, W_{m(n+1)-1}$ by
  - $W_i = K(-, i)$, for any $0 \leq i \leq k-1$
  - $W_i = W_{i-k} \oplus T(W_{i-1})$, where:
    - $T(W) = \begin{cases} SB(RB(W)) \oplus Rcon(i/k), & \text{if } i \ mod \ k = 0 \\ SB(W), & \text{if } k > 6 \text{ and } i \ mod \ k = 4 \\ W, & \text{otherwise} \end{cases}$
    - $RB((z_0, z_1, z_2, z_3)^t) = (z_1, z_2, z_3, z_0)^t$
    - $SB((z_0, z_1, z_0, z_3)^t) = (S(z_0), S(z_1), S(z_2), S(z_3))^t$
    - $Rcon(i) = (RC(i), (00)_h, (00)_h, (00)_h)$, where $RC(1) = (01)_h$ and $RC(i) = x \bullet RC(i-1)$, for all $k \leq i \leq m(n+1)-1$
- $K_i = (W_{im}, W_{im+1}, \ldots, W_{(i+1)m-1})$, for any $i \geq 0$

AES correctness

Proposition 4

For any $k, m \in \{4, 6, 8\}$, $K \in \mathcal{M}_{4\times k}(\mathbb{Z}_2^8)$, and $Z, X \in \mathcal{M}_{4\times m}(\mathbb{Z}_2^8)$, the following properties hold:

1. M_1 *is invertible and, therefore, the transformation* S *is invertible.*
2. M_2 *is invertible and, therefore, the transformation* Mc *is invertible.*
3. $A_Z \circ A_Z = Id$.
4. $Sh \circ S = S \circ S_h$.
5. $Mc^{-1} \circ A_Z = A_{Mc^{-1}(Z)} \circ Mc^{-1}$.
6. $T_Z^i \circ T_Z^f = Sh \circ S$.
7. $T_Z^{-1} \circ Sh \circ S \circ T_Z = Sh \circ S$.
8. $T_Z^{-f} \circ Sh \circ S \circ T_Z^i = Id$.
9. $d_K(e_K(X)) = X$.

AES security

1. 2002 : Courtois and Pieprzyk showed that AES can be written as an over-defined system of multivariate quadratic equations (MQ). For example authors showed that for 128-bit Rijndael, the problem of recovering the secret key from one single plaintext can be written as a system of 8000 quadratic equations with 1600 binary unknowns
2. 2009, Biryukov, Khovratovich : key-related attack
   2.1 Time $2^{99.5}$
3. 2011, Bogdanov, Khovratovich, Rechberger : key recovery attack
   3.1 For AES-128, it is faster than brute force by a factor of about four
   3.2 For AES-192 and AES-256, $2^{190.2}$ and $2^{254.6}$ operations are needed for key recovery attack

PRF

Random functions

1. The term "random function" is misleading
2. The randomness of a function refers to the way it was chosen from a given set of functions
3. "Randomness" is not an attribute of a function
4. "Random function" = function chosen at random

Families of functions

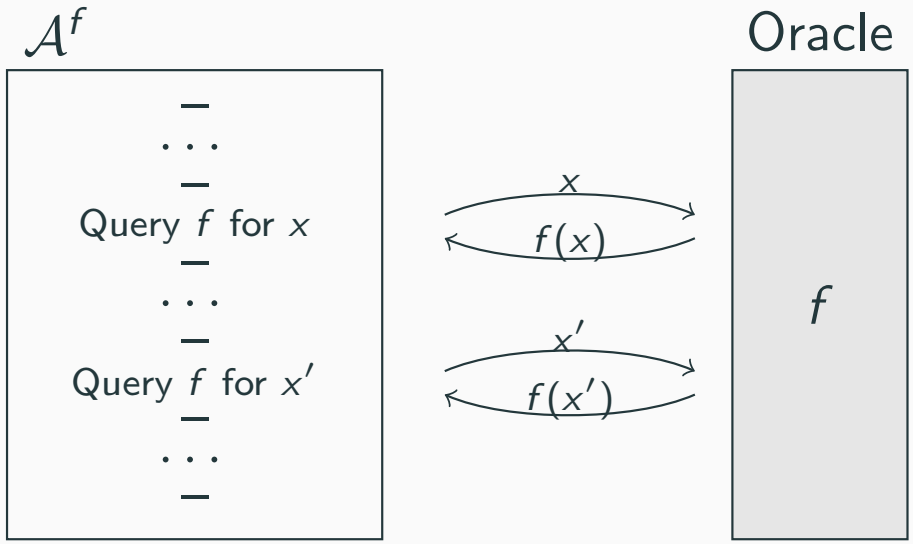
1. A pseudo-random function is a family of functions with the property that if we randomly choose a function from this family then its input-output behavior is computationally indistinguishable from that of a random function
2. Family of functions from $\{0,1\}^{\ell_1(n)}$ to $\{0,1\}^{\ell_2(n)}$, where ℓ_1 and ℓ_2 are polynomials with positive values:
   - 2.1 $F_{\ell_1,\ell_2,n}$: random variable with values in $(\{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)})$
   - 2.2 $F_{\ell_1,\ell_2} = (F_{\ell_1,\ell_2,n})_{n\in\mathbb{N}}$
3. Special cases :
   - 3.1 $H_{\ell_1,\ell_2} = (H_{\ell_1,\ell_2,n})_{n\in\mathbb{N}}$ is the uniform distribution
   - 3.2 $H^0_\ell = (H^0_{\ell,n})_{n\in\mathbb{N}}$ is the uniform distribution on all permutations on $\{0,1\}^{\ell(n)}$, $n \in \mathbb{N}$

We will avoid the subscript "ℓ_1, ℓ_2" or "ℓ" whenever these polynomials are clear from context

Algorithms with oracle access to a function

Given a function $f : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$, the notation $\mathcal{A}^f$ signifies an algorithm that has oracle access to the function f in the following sense:

1. The algorithm can adaptively query an oracle for f in the sense that on an input x it gets $f(x)$
2. The algorithm has only a "black-box" (input-output) view on f in the sense that it does not know how the function f is evaluated



Oracle indistinguishability

We use $\mathcal{A}^\circ$ to denote algorithms with oracle access to functions; $\mathcal{A}^f$ is an instantiation of the oracle by f.

Definition 5 (Oracle indistinguishability)

Two families F and G of functions from $\{0,1\}^{\ell_1(n)}$ to $\{0,1\}^{\ell_2(n)}$ are called computationally indistinguishable if, for any PPT algorithm $\mathcal{A}^\circ$ with oracle access to functions from $\{0,1\}^{\ell_1(n)}$ to $\{0,1\}^{\ell_2(n)}$, its advantage

$$\begin{aligned} Adv^{prf}_{\mathcal{A}^\cdot,F,G}(n) &= |P(1 \leftarrow \mathcal{A}^f(1^n) : f \leftarrow F_{\ell_1,\ell_2,n}) - \\ &\quad P(1 \leftarrow \mathcal{A}^f(1^n) : f \leftarrow G_{\ell_1,\ell_2,n})| \end{aligned}$$

is negligible.

Remark 6

Oracle indistinguishability enjoys all the properties we have already proved for computational indistinguishability.

Pseudo-random functions

Definition 7 (Pseudo-random function)

A set of functions

$$\mathcal{F} = \{f_K : \{0,1\}^{\ell_1(|K|)} \to \{0,1\}^{\ell_2(|K|)} \mid K \in \{0,1\}^*\}$$

is called pseudo-random if it is:

1. Efficiently computable : there exists a deterministic polynomial-time algorithm that on input $K \in \{0,1\}^*$, and $x \in \{0,1\}^{\ell_1(|K|)}$ returns $f_K(x)$;
2. Pseudo-random : the family $F = (F_n)_{n \in \mathbb{N}}$, where F_n is a random variable uniformly distributed over (the multi-set) $\{f_K \in \mathcal{F} \mid |K| = n\}$, is computationally indistinguishable from H_{ℓ_1,ℓ_2}.

Pseudo-random functions

Remark 8

Why "pseudo-random function"? Because $\mathcal{F}$ can be defined as a (partial) function

$$\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$

f_K can be regarded as the encryption function with the key K.

$\mathcal{F}$ is usually given as a family of functions

$$\mathcal{F} = (\mathcal{F}_n)_{n \in \mathbb{N}},$$

where $\mathcal{F}_n = \{f_K : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)} \mid K \in \{0,1\}^n\}$. The uniform distribution on $\{0,1\}^n$ induces a uniform distribution on $\mathcal{F}_n$ (it is consider that $\mathcal{F}_n$ contains 2^n functions, although we may have $f_{K_1} = f_{K_2}$ for some $K_1 \neq K_2$).

Construction of PRF

Case 1 : $\ell_1(n) = \ell_2(n) = n$

1. Let G be a deterministic algorithm that expands n-bit inputs into $2n$-bit inputs
2. Let $G(K) = G_0(K)G_1(K)$, where $|G_0(K)| = |G_1(K)| = |K| = n$
3. Define $f_K : \{0,1\}^n \to \{0,1\}^n$ by

$$f_K(b_1 b_2 \cdots b_n) = G_{b_n}(\cdots(G_{b_2}(G_{b_1}(K)))\cdots)$$

   for all $b_1 b_2 \cdots b_n \in \{0,1\}^n$
4. Define F_n be the random variable that outputs f_K when K is uniformly at random chosen from $\{0,1\}^n$.

Theorem 9

If G is a PRG, then $F = (F_n)_{n\in\mathbb{N}}$ defined as above is PRF.

Construction of PRF

Case 2 : $n \leq \ell_1(n) \leq \ell_2(n)$

1. Let G, G_0, G_1 as in Case 1
2. Let G' be a deterministic algorithm that expands n-bit inputs into $\ell_2(n)$-bit inputs
3. Define $f_K : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ by

$$f_K(b_1 b_2 \cdots b_{\ell_1(n)}) = G'(G_{b_{\ell_1(n)}}(\cdots(G_{b_2}(G_{b_1}(K)))\cdots)$$

   for all $b_1 b_2 \cdots b_{\ell_1(n)} \in \{0,1\}^n$
4. Define F_n as in Case 1

Theorem 10

If G is a PRG and G' is polynomial-time computable and $(G'(U_n))_{n\in\mathbb{N}}$ is pseudo-random, then $F = (F_n)_{n\in\mathbb{N}}$ defined as above is PRF.

Block ciphers from PRF

Cipher 1 (SKE cipher from PRF)

Let F be a PRF with $\ell_1(n) = \ell_2(n) = n$. Define an SKE scheme $\mathcal{S}(G) = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ over $(\{0,1\}^n, \{0,1\}^n, \{0,1\}^n)$ by:

1. $\mathcal{G}(\lambda)$ *: output* $K \leftarrow \{0,1\}^\lambda$
2. $\mathcal{E}(K, m)$ *: choose* $r \leftarrow \{0,1\}^\lambda$ *and output*

$$c = (r, m \oplus f_K(r))$$

3. $\mathcal{D}(K, c)$ *: output* $m = s \oplus f_K(r)$*, where* $c = (r, s)$

Theorem 11

The SKE scheme in Description 1 is IND-CPA, provided that F is a PRF.

PRP

Weak pseudo-random permutations

Definition 12 (Weak pseudo-random permutation (weak PRP))

A set of permutations

$$\mathcal{P} = \{f_K : \{0,1\}^{\ell(|K|)} \to \{0,1\}^{\ell(|K|)} \mid K \in \{0,1\}^*\}$$

is called pseudo-random if it is:

1. Efficiently computable :
   - 1.1 there exists a deterministic polynomial-time algorithm that on input $K \in \{0,1\}^*$, and $x \in \{0,1\}^{\ell(|K|)}$ returns $f_K(x)$;
   - 1.2 there exists a deterministic polynomial-time algorithm that on input $K \in \{0,1\}^*$, and $y \in \{0,1\}^{\ell(|K|)}$ returns $f_K^{-1}(y)$;
2. Pseudo-random : the family $P = (P_n)_{n \in \mathbb{N}}$, where P_n is a random variable uniformly distributed over (the multi-set) $\{f_K \in \mathcal{P} \mid |K| = n\}$, is computationally indistinguishable from H_ℓ^0.

Strong pseudo-random permutations

Definition 13 (Strong pseudo-random permutation (strong PRP))

A set of permutations

$$\mathcal{P} = \{f_K : \{0,1\}^{\ell(|K|)} \to \{0,1\}^{\ell(|K|)} \mid K \in \{0,1\}^*\}$$

is called pseudo-random if it is:

1. Efficiently computable :
   - 1.1 there exists a deterministic polynomial-time algorithm that on input $K \in \{0,1\}^*$, and $x \in \{0,1\}^{\ell(|K|)}$ returns $f_K(x)$;
   - 1.2 there exists a deterministic polynomial-time algorithm that on input $K \in \{0,1\}^*$, and $y \in \{0,1\}^{\ell(|K|)}$ returns $f_K^{-1}(y)$;
2. Pseudo-random : the family $P = (P_n)_{n \in \mathbb{N}}$, where P_n is a random variable uniformly distributed over (the multi-set) $\{f_K \in \mathcal{P} \mid |K| = n\}$, is computationally indistinguishable from H_ℓ^0 by PPT algorithms that have oracle access to both f and f^{-1}.

Pseudo-random permutations

Remark 14

1. *Strong PRP are simply referred to as PRP*
2. *PRP are sometimes called block ciphers*

Example 15 (PRP candidates)

1. $DES : \{0,1\}^{56} \times \{0,1\}^{64} \to \{0,1\}^{64}$
2. $3DES : \{0,1\}^{168} \times \{0,1\}^{64} \to \{0,1\}^{64}$
3. $AES\text{-}128 : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$

PRP and PRF

Proposition 16

Any PRP is also a PRF

Definition 17

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a function. The Feistel function associated to f is the function $D_f : \{0,1\}^{2n} \to \{0,1\}^{2n}$ given by

$$D_f(uv) = v \oplus f(u)$$

for any $u, v \in \{0,1\}^n$.

Definition 18

A k-round Feistel function is a function of the form $D_f \circ \cdots \circ D_f$, where $k \geq 1$, D_f is a Feistel function, and the composition is iterated k times.

PRP and PRF

Proposition 19

1- and 2-round Feistel functions are not PRP no matter how the function f on which they are constructed is.

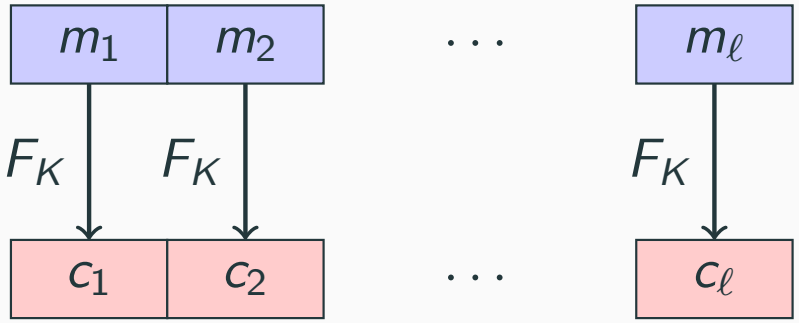
Theorem 20 (Luby-Rackoff)

1. *3-round Feistel functions based on PRF are weak PRP.*
2. *4-round Feistel functions based on PRF are (strong) PRP.*

Modes of operations

Electronic Code Block (ECB)

$F = (F_n)_{n \in \mathbb{N}}$ is a PRP



Theorem 21

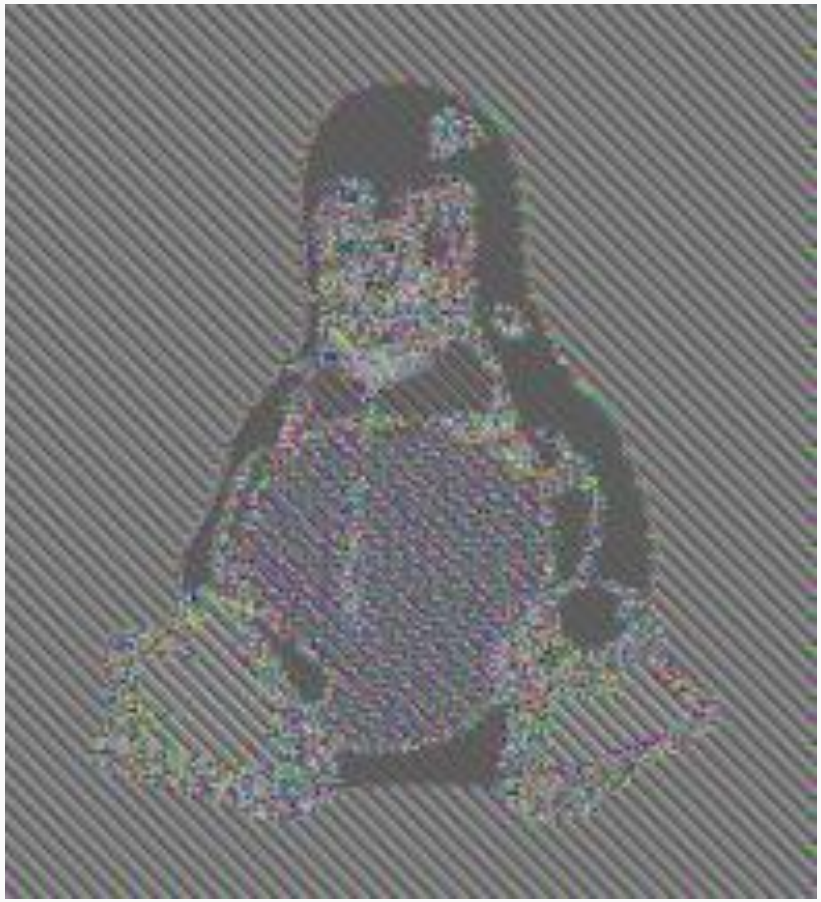
ECB is not IND-KPA.

Proof.

$m_i = m_j \quad \Leftarrow \quad c_i = c_j.$ □

ECB illustrated

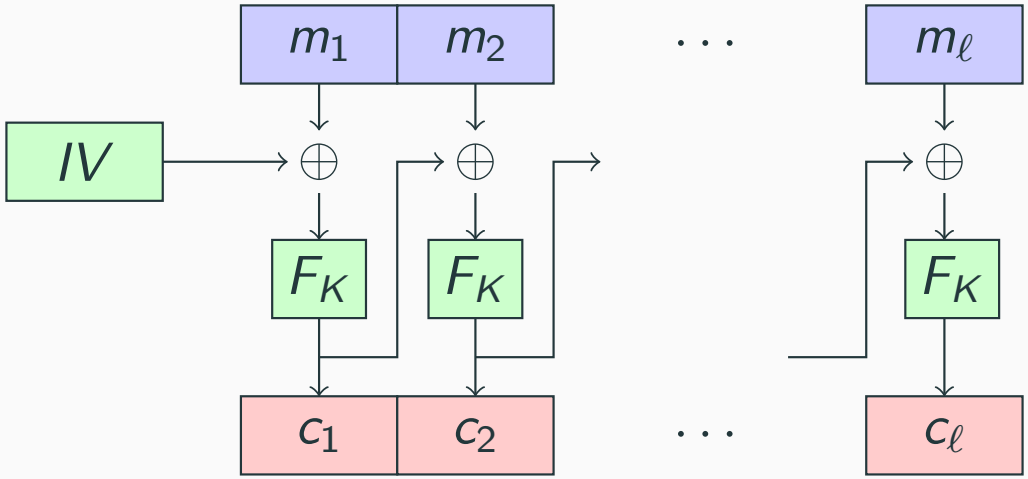
Original image



ECB encryption

Cipher Block Chaining (CBC)

$F = (F_n)_{n \in \mathbb{N}}$ is a PRP



- $c_0 = IV$
- $c_i = F_K(m_i \oplus c_{i-1})$, for all $i \geq 1$
- ciphertext : $(IV, c_1 \cdots c_\ell)$

Cipher Block Chaining (CBC)

Theorem 22

If $F = (F_n)_{n\in\mathbb{N}}$ is a PRP, then CBC with F is IND-CPA.

Proof.

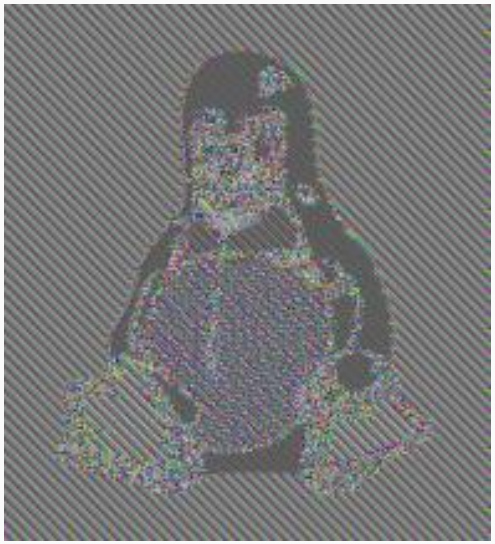
Show that for any adversary $\mathcal{A}$ against the scheme $\mathcal{S}$, there exists a PRP adversary $\mathcal{B}$ such that

$$Adv^{\mathsf{dctr}}_{\mathcal{A},\mathcal{S}}(\lambda) \leq 2 \cdot Adv^{\mathsf{prp}}_{\mathcal{B},F}(\lambda) + \frac{q(\lambda)^2\ell^2}{2^{\lambda-1}},$$

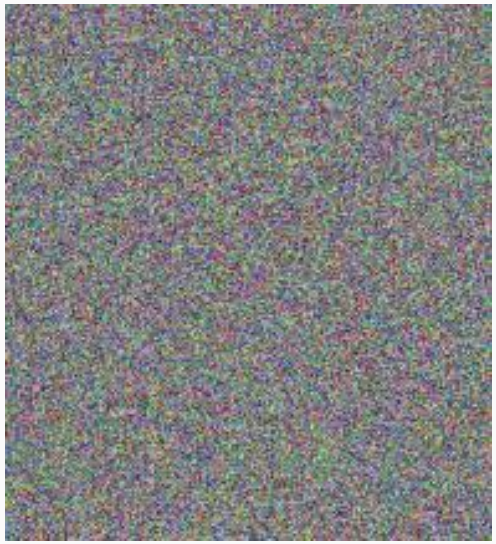
where ℓ is the input block-length of the messages and $q(\lambda)$ is the maximum number of queries $\mathcal{A}$ makes to its challenger. □

CBC versus ECB

Original image



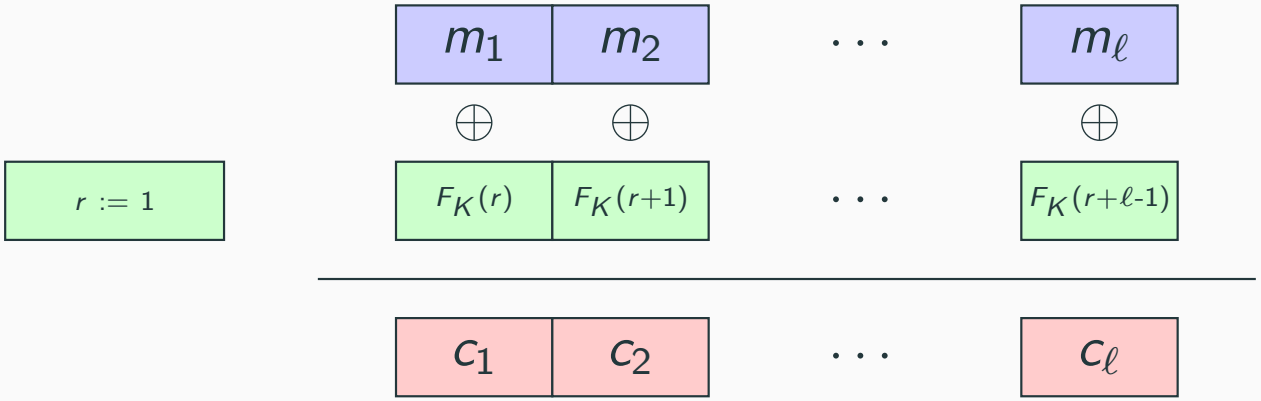
ECB encryption



CBC encryption

Deterministic counter mode (DCTR)

$F = (F_n)_{n \in \mathbb{N}}$ is a PRF



Remark 23

1. *The ciphertext is $c_1 \cdots c_\ell$ (assuming r is publicly known)*
2. *The scheme works like a stream cipher with the PRG G given by*

$$G(K) = F_K(1) \| F_K(2) \| \cdots \| F_K(\ell)$$

3. *DES, 3DES, AES can be used with such a construction*

Deterministic counter mode (DCTR)

Theorem 24

If $F = (F_n)_{n\in\mathbb{N}}$ is a PRF, then DCTR with F is IND-KPA but not IND-CPA.

Proof.

For IND-CPA: query for $m_1 m_2$ and $m_3 m_4$, and then request challenge for $(m_1 m_4, m_3 m_2)$.

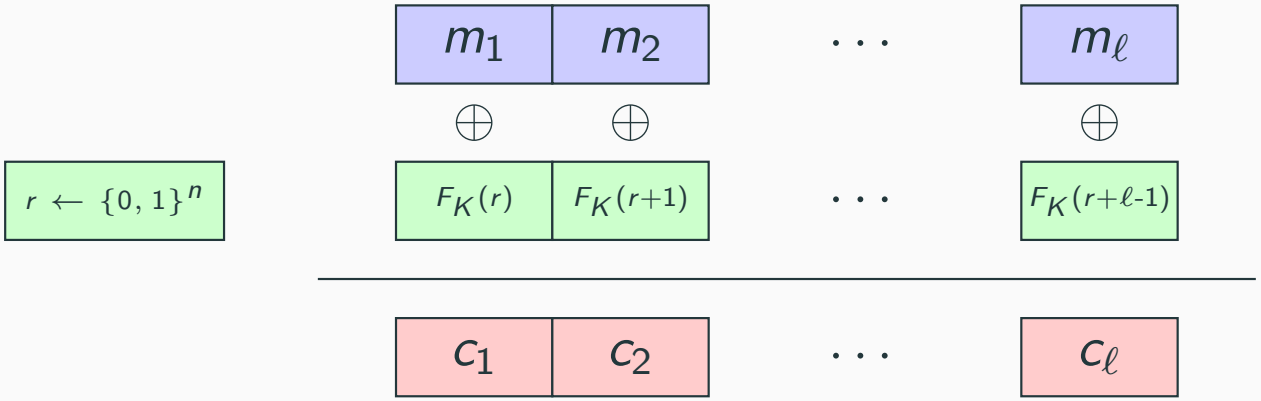
For IND-KPA: show that for any adversary $\mathcal{A}$ against the scheme $\mathcal{S}$, there exists a PRF adversary $\mathcal{B}$ such that

$$Adv^{\mathsf{dctr}}_{\mathcal{A},\mathcal{S}}(\lambda) = 2 \cdot Adv^{\mathsf{prf}}_{\mathcal{B},F}(\lambda)$$

□

Counter mode (CTR)

$F = (F_n)_{n \in \mathbb{N}}$ is a PRF



Remark 25

1. *The ciphertext is* $(r, c_1 \cdots c_\ell)$
2. *The scheme is similar to DCTR expect for the fact that the counter* r *is uniformly at random generated from* $\{0,1\}^n$
3. *DES, 3DES, AES can be used with such a construction*

Counter mode (CTR)

Theorem 26

If $F = (F_n)_{n\in\mathbb{N}}$ is a PRF, then CTR with F is IND-CPA.

Proof.

Show that for any adversary $\mathcal{A}$ against the scheme $\mathcal{S}$, there exists a PRF adversary $\mathcal{B}$ such that

$$Adv^{\mathsf{dctr}}_{\mathcal{A},\mathcal{S}}(\lambda) \leq 2 \cdot Adv^{\mathsf{prf}}_{\mathcal{B},F}(\lambda) + \frac{q(\lambda)^2\ell}{2^{\lambda-1}},$$

where ℓ is the input block-length of the messages and $q(\lambda)$ is the maximum number of queries $\mathcal{A}$ makes to its challenger. □

Output feedback (OFB) and cipher feedback (CFB)

1. The key stream in CTR mode is

$$F_K(r) \parallel F_K(r+1) \parallel F_K(r+2) \parallel \cdots$$

   where $r \leftarrow \{0,1\}^n$

2. The OFB and CFB modes are defined as the CTR mode but with a different key stream generation :

   2.1 The key stream in OFB mode is

$$F_K(r) \parallel F_K(F_K(r)) \parallel F_K(F_K(F_K(r))) \parallel \cdots$$

   where $r \leftarrow \{0,1\}^n$

   2.2 The key stream in CFB mode is

$$F_K(r) \parallel F_K(c_1) \parallel F_K(c_2) \parallel \cdots$$

   where $r \leftarrow \{0,1\}^n$

Reading and exercise guide

Reading and exercise guide

Course readings:

1. Pages 70-98 from [1].

References

[1] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

Hash Functions

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Hash functions

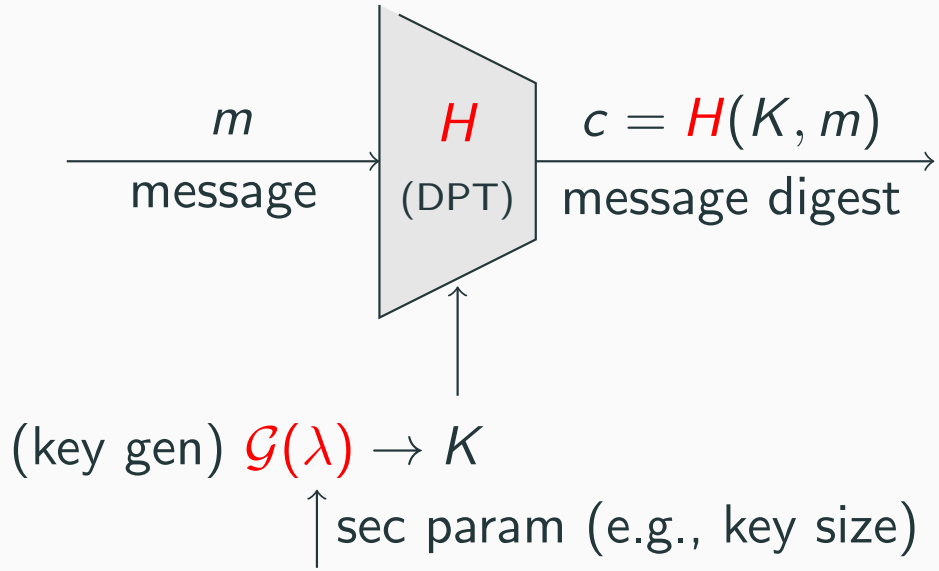
Introduction

A hash function outputs a fixed-length bitstring (e.g., 128, 160, 224, 256, 384, 512) when applied to an arbitrary-length bitstring.

Hash functions are used in many cryptographic applications such as:

- signing messages, in connection with digital signatures (signing a document should be a fast operation and the signature should be small so that it can be put on a smart card);
- identifying files on peer-to-peer file sharing networks;
- ensuring security of micro-payment schemes (e.g., PayWord);
- etc.

Hash function: pictorial view



$\mathcal{H} = (\mathcal{G}, H)$ - (keyed) hash function

When no key is used, H is called a hash function.

Hash function: formal definition

Definition 1

A keyed hash function over $(\mathcal{K}, X, Z)$ is a pair of algorithms $\mathcal{H} = (\mathcal{G}, H)$ such that:

1. $\mathcal{G}$ is a PPT algorithm which takes as input a security parameter λ and outputs a key $K \in \mathcal{K}$;
2. H is a DPT algorithm which takes as input a key K and a message $m \in X$ and outputs a digest $H(K, m) \in Z$.

As usual, X and Z are sets of binary strings; typically, $Z = \{0,1\}^{\ell}$ for some small ℓ (e.g., 128 or 256).

When $|\mathcal{K}| = 1$, H is simply written as a function from X into Z and it is called a hash function.

Security of hash functions

Security of hash functions

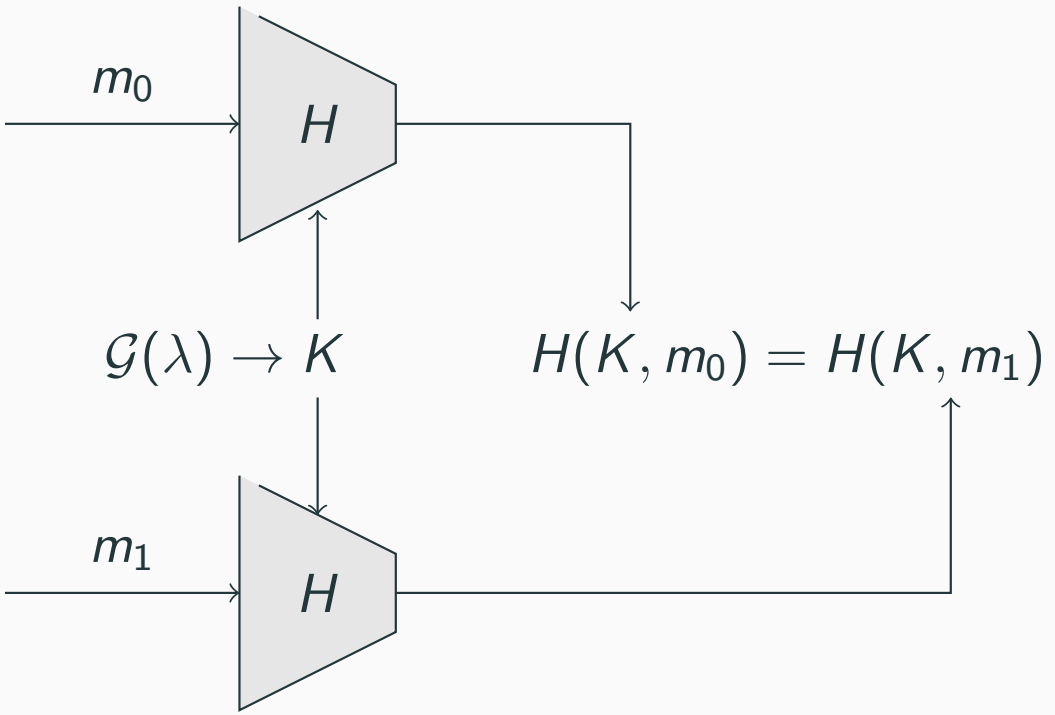
The security of a hash function can be seen from different angles. In this section we will distinguish between:

1. Collision-resistant hash functions;
2. Universal hash functions;
3. One-way hash functions (also known as pre-image resistant hash functions);
4. Universal one-way hash functions (also known as second pre-image resistant hash function).

Sometimes, hash functions are asked to be pseudo-random in the sense that it should be hard to distinguish between their output and the output of a pseudo-random generator.

A hash function has provable security if its security can be proven based on the assumption that some mathematical problem is hard.

Collision-resistant hash functions: pictorial view



If $m_0 \neq m_1$ then
(m_0, m_1) is a collision

A keyed hash function $\mathcal{H}$ is collision-resistant (CRHF) if no adversary, having the key K, can compute a collision (m_0, m_1) for H under K with a higher than negligible probability.

Collision-resistant hash functions: formal definition

A collision for H under K is any pair (m_0, m_1) of distinct messages such that $H(K, m_0) = H(K, m_1)$.

Experiment $CRHF^{kka}_{\mathcal{A},\mathcal{H}}(\lambda)$

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$ and gives it to $\mathcal{A}$
2: The adversary $\mathcal{A}$ generates a pair of messages (m_0, m_1)
3: If (m_0, m_1) is a collision of H under K then return 1, else return 0.

Remark that the adversary mounts a known-key attack (kka).

The advantage of $\mathcal{A}$ is $Adv^{cr\text{-}kka}_{\mathcal{A},\mathcal{H}}(\lambda) = P(CRHF^{kka}_{\mathcal{A},\mathcal{H}}(\lambda) = 1)$

Definition 2

A keyed hash function $\mathcal{H}$ is collision-resistant (CRHF) if $Adv^{cr\text{-}kka}_{\mathcal{A},\mathcal{H}}(\lambda)$ is negligible for all PPT algorithms $\mathcal{A}$.

Universal hash functions

Universal hash functions are defined similarly to collision-resistant hash functions, but with the difference that the key is not given to the adversary.

Experiment $UHF_{\mathcal{A},\mathcal{H}}(\lambda)$

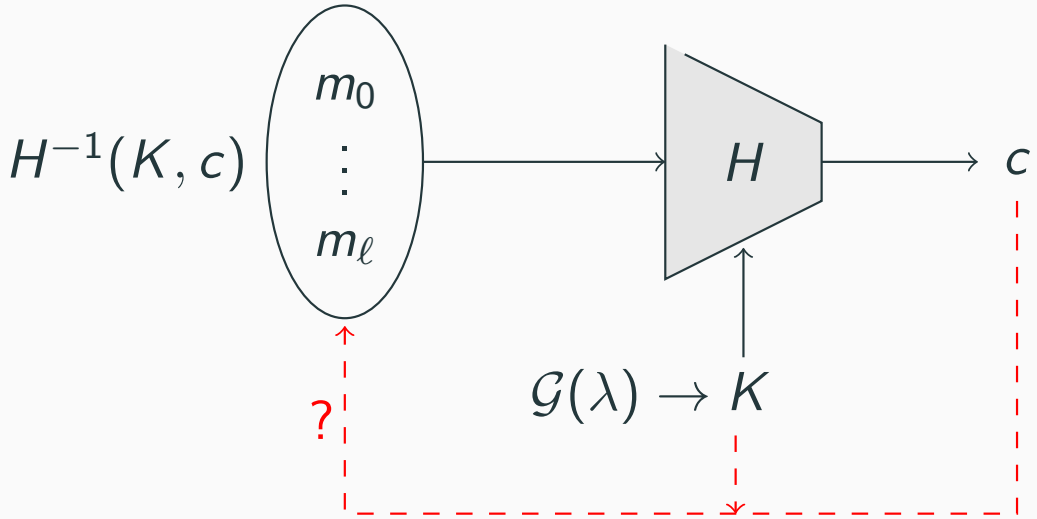
1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$
2: The adversary $\mathcal{A}$ generates a pair of messages (m_0, m_1)
3: If (m_0, m_1) is a collision of H under K then return 1, else return 0.

The advantage of $\mathcal{A}$ is $Adv^{u}_{\mathcal{A},\mathcal{H}}(\lambda) = P(UHF_{\mathcal{A},\mathcal{H}}(\lambda) = 1)$

Definition 3

A keyed hash function $\mathcal{H}$ is universal (UHR) if $Adv^{u}_{\mathcal{A},\mathcal{H}}(\lambda)$ is negligible for all PPT algorithms $\mathcal{A}$.

One-way hash functions: pictorial view



A keyed hash function $\mathcal{H}$ is one-way (OWHF) if no adversary, given a randomly generated key K and a message digest c obtained with K, can compute $m \in H^{-1}(K, c)$ with a higher than negligible probability.

One-way hash functions: formal definition

Experiment $OWHF^{kka}_{\mathcal{A},\mathcal{H}}(\lambda)$

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$ and y in the range of H_K, and gives them to $\mathcal{A}$
2: The adversary $\mathcal{A}(K, y)$ generates m
3: If $H(K, m) = y$ then return 1, else return 0.

The adversary must compute a collision for a message in $H^{-1}(K, y)$!

The advantage of $\mathcal{A}$ is $Adv^{ow\text{-}kka}_{\mathcal{A},\mathcal{H}}(\lambda) = P(OWHF^{kka}_{\mathcal{A},\mathcal{H}}(\lambda) = 1)$

Definition 4

A keyed hash function $\mathcal{H}$ is one-way (OWHF) if $Adv^{ow\text{-}kka}_{\mathcal{A},\mathcal{H}}(\lambda)$ is negligible for all PPT algorithms $\mathcal{A}$.

Universal one-way hash functions

Universal one-way functions are defined similarly to one-way functions, but with the difference that the adversary must compute a collision for a message chosen by him.

Experiment $UOWHF_{\mathcal{A},\mathcal{H}}(\lambda)$

1: The adversary $\mathcal{A}$ generates a message m_0
2: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$ and gives it to $\mathcal{A}$
3: The adversary $\mathcal{A}(K, m_0)$ generates m_1
4: If (m_0, m_1) is a collision of H under K then return 1, else return 0.

The advantage of $\mathcal{A}$ is $Adv^{uow\text{-}kka}_{\mathcal{A},\mathcal{H}}(\lambda) = P(UOWHF_{\mathcal{A},\mathcal{H}}(\lambda) = 1)$

Definition 5

A keyed hash function $\mathcal{H}$ is universal one-way (UOWHF) if $Adv^{uow\text{-}kka}_{\mathcal{A},\mathcal{H}}(\lambda)$ is negligible for all PPT algorithms $\mathcal{A}$.

Relationship between CRHF, UOWHF, and UHF

Theorem 6

Let $\mathcal{H}$ be a hash function. Then:

1. *For any adversary $\mathcal{A}$ there exists an adversary $\mathcal{B}$ having the same running time as $\mathcal{A}$ such that*

$$Adv^{uow}_{\mathcal{A},\mathcal{H}}(\lambda) \leq Adv^{cr\text{-}kka}_{\mathcal{B},\mathcal{H}}(\lambda)$$

2. *For any adversary $\mathcal{A}$ there exists an adversary $\mathcal{B}$ having the same running time as $\mathcal{A}$ such that*

$$Adv^{u}_{\mathcal{A},\mathcal{H}}(\lambda) \leq Adv^{uow}_{\mathcal{B},\mathcal{H}}(\lambda)$$

Corollary 7

Any CRHF is also a UOWHF, and any UOWHF is also a UHF.

Relationship between UOWHF and OWHF

Theorem 8

Let $\mathcal{H}$ be a hash function. Then, for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{B}$ such that

$$Adv^{ow\text{-}kka}_{\mathcal{A},\mathcal{H}}(\lambda) \leq Adv^{uow\text{-}kka}_{\mathcal{B},\mathcal{H}}(\lambda) +$$

$$P(|H(K,y)^{-1}| = 1 \,:\, y = H(K,x),\, K \leftarrow \mathcal{K},\, x \leftarrow X)$$

Moreover, the running time of $\mathcal{B}$ is that of $\mathcal{A}$ plus the time to sample an element from X and compute H once.

Corollary 9

Any UOWHF is also a OWHF, as long as the domain of the hash function is significantly larger than its range.

Finding collisions when inversability is easy

Let $H(K, \cdot) : X \to Z$ be a hash function such that $|X| \geq 2|Z|$. If there exists an algorithm $\mathcal{A}$ to invert easily H, then there exists a PPT algorithm $\mathcal{B}$ that finds a collision for H with probability at least $1/2$.

Algorithm $\mathcal{B}$
input: X, Z, H, and K;
output: a collision for H or nothing;
begin
1. choose $m_0 \in X$;
2. $\mathcal{B}$ gets a key K;
3. $y := H(K, m_0)$;
4. $m_1 := \mathcal{A}(K, y)$;
5. if $m_1 \neq m_0$ then (m_0, m_1) is a collision else quit.

end

Finding collisions when inversability is easy

$\mathcal{B}$'s probability of success is the probability of choosing $m_0 \in X$ and $m_1 \in [m_0]_{Ker(H(K,\cdot))}$ such that $m_1 \neq m_0$

$$\begin{aligned}
P(succes) &= \frac{\sum_{m_0 \in X} \frac{|[m_0]|-1}{|[m_0]|}}{|X|} &&= \frac{1}{|X|}\sum_{c \in C}\sum_{m_0 \in c}\frac{|c|-1}{|c|} \\
&= \frac{1}{|X|}\sum_{c \in C}(|c|-1) &&= \frac{1}{|X|}(\sum_{c \in C}|c| - \sum_{c \in C}1) \\
&\geq \frac{1}{|X|}(|X|-|Z|) &&\geq \frac{|X|-\frac{|X|}{2}}{|X|} \\
&= \frac{1}{2}
\end{aligned}$$

(C is the set of all equivalence classes induced by $ker(H(K,\cdot))$. We have also used the property $|X| \geq 2|Z|$).

The random oracle model for hash functions

The random oracle model is a way to analyze schemes that need a hash function by replacing the hash function with some black box (the random oracle) which evaluates the function as follows:

- If you give it an input it hasn't seen yet, it gives you an output selected uniformly at random from its possible outputs. It then saves that output for later use;
- If you give it an input it has seen before, it gives you the output it saved previously.

No actual hash function is a random oracle!

Unfortunately, replacing a hash function in a cryptographic scheme with a random oracle can conclude that the scheme is secure when it is not (by using the hash function). So, random oracles are an imperfect model, and we generally prefer security proofs not relying on them.

The bithday attack

Finding collisions: the birthday attack

The birthday attack is based on the birthday paradox which says that in a group of 23 (randomly chosen) people, at least two of them will share a birthday with probability at least $1/2$.

Given a group of r people, the probability that no two people share a birthday is

$$p_{365,r} = \frac{365}{365} \cdot \frac{364}{365} \cdots \frac{365 - r + 1}{365} = \frac{365!}{(365 - r)! \cdot 365^r}$$

Therefore, the probability that at least two people share a birthday is

$$1 - p_{365,r}$$

For $r = 23$, $1 - p_{365,r} \geq 1/2$.

A generalization of the birthday paradox

We have m properties and r objects, each object having exactly one of the m properties. The probability that at least two objects share the same property is $1 - p_{m,r}$, where

$$p_{m,r} = \frac{m!}{(m-r)! \cdot m^r} = \left(1 - \frac{1}{m}\right) \cdots \left(1 - \frac{r-1}{m}\right)$$

Lemma 10

Let m and r be natural numbers such that $\lfloor \sqrt{2cm} \rfloor < r < m$, where $c > 0$ is a real constant. Then,

$$1 - p_{m,r} > 1 - e^{-c}$$

Example 11

$1 - p_{m,r} > 1 - e^{-c} \geq \frac{1}{2}$ for $c \geq \ln 2 \sim 0.693$ and $m > r > \lfloor \sqrt{2cm} \rfloor$.

The birthday paradox applied to hash functions

The birthday paradox is used to attack hash functions as follows:

- m = number of possible message digests;
- r = number of messages for which a message digest will be computed;
- if $\lfloor\sqrt{2cm}\rfloor < r < m$ for some real constant $c > 0$, then $1 - p_{m,r} > 1 - e^{-c}$.

Example 12

Let $m = 2^{40}$ and r such that $2^{40} > r > \lfloor 2^{20}\sqrt{2\ln 2}\rfloor \approx 1.200.000$.

The probability of getting a collision is greater than $1/2$. Therefore, 40-bit message digests do not ensure security!

For 128-bit message digests, the attack needs to compute at least 2^{64} message digests to get a collision with the probability at least $1/2$.

Construction of CRHFs

Construction of CRHFs

Two practical classes of techniques for constructing hash functions:

1. Based on compression functions, like the the Merkle-Damgard (MD) transform;
2. Based on permutations, like the sponge construction.

Compression functions

A compression function is a function that transforms one or more fixed-length inputs into a fixed-length output. Moreover, the length of the output is smaller than the length of some input.

Constructing compression functions (just a selection):

1. From block ciphers:
   - 1.1 Diffie-Hellman, 1976: $H_i = E_{H_{i-1}, m_i}(const)$
   - 1.2 Rabin, 1978: $H_i = E_{m_i}(H_{i-1})$
   - 1.3 Davies-Meyer: $H_i = E_{m_i}(H_{i-1}) \oplus H_{i-1}$
   - 1.4 Matyas-Meyer-Oseas: $H_i = E_{H_{i-1}}(m_i) \oplus m_i$
   - 1.5 Miyagucchi-Preneel: $H_i = E_{H_{i-1}}(m_i) \oplus m_i \oplus H_{i-1}$
2. Based on hardness assumptions:
   - 2.1 Chaum-van_Heijst-Pfitzmann, 1991

Davies-Meyer's compression function

Davies-Meyer's compression function

$$H_i = E_{m_i}(H_{i-1}) \oplus H_{i-1}$$

- The functions is iterated on the number of block messages;
- i is the current iteration to be performed;
- H_{i-1} is the output value at the previous iteration. H_0 is an initialization value;
- E_{m_i} denotes a block encryption with key m_i, where m_i is the current message block.

The security of this compression function is tackled in [2].

Chaum-van_Heijst-Pfitzmann's compression function

Chaum-van_Heijst-Pfitzmann compression function

- let p and q be primes such that $p = 2q + 1$;
- let $\alpha, \beta \in \mathbb{Z}_p^*$ be primitive elements (and let $\beta = \alpha^c \ mod \ p$, for some c);
- $h : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{Z}_p^*$ is given by

$$h(x_1, x_2) = \alpha^{x_1} \beta^{x_2} \ mod \ p,$$

for any $x_1, x_2 \in \mathbb{Z}_q$.

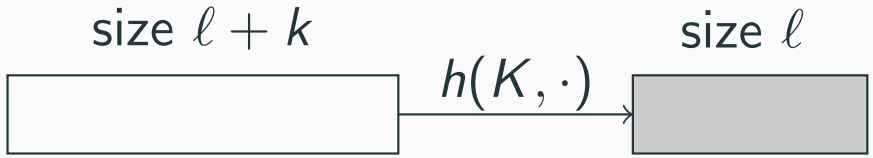
Theorem 13

If a collision of h can be computed efficiently, then c can be computed efficiently.

According to Theorem 13, h is collision-resistant if DLP is intractable.

The MD transform

- Use a compression function $h : \mathcal{K} \times \{0,1\}^{\ell+k} \to \{0,1\}^{\ell}$

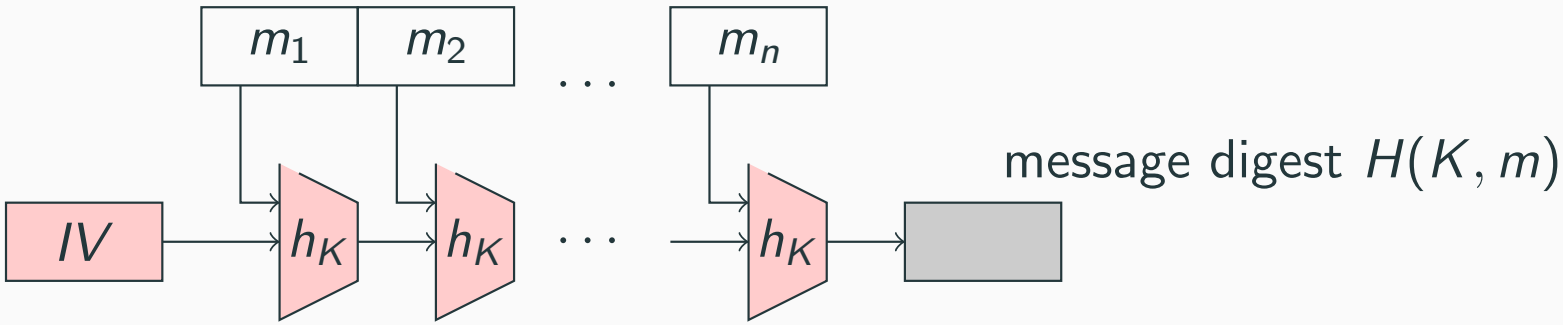


- Use an MD-complaint padding $pad : \{0,1\}^{<2^{\ell}} \to \bigcup_{n \geq 1} \{0,1\}^{n\ell}$ with the following properties:

  1. m is a prefix of $pad(m)$;
  2. if $|m_1| = |m_2|$ then $|pad(m_1)| = |pad(m_2)|$;
  3. if $m_1 \neq m_2$, then the last block of $pad(m_1)$ is different than the last block of $pad(m_2)$.

The MD transform

- Iterate h on messages m as follows:
  1. $pad(m) = m_1 \parallel \cdots \parallel m_n$ with $|m_i| = k$ for all i
  2. $V := IV$, where $IV \leftarrow \{0,1\}^{\ell}$
  3. for $i := 1$ to n do $V := h(K, m_i \parallel V)$
  4. return V



Theorem 14

If h is collision-resistant, then the MD-transform based on h is so.

The MD transform in practice

Practical hash functions based on the MD-transform:

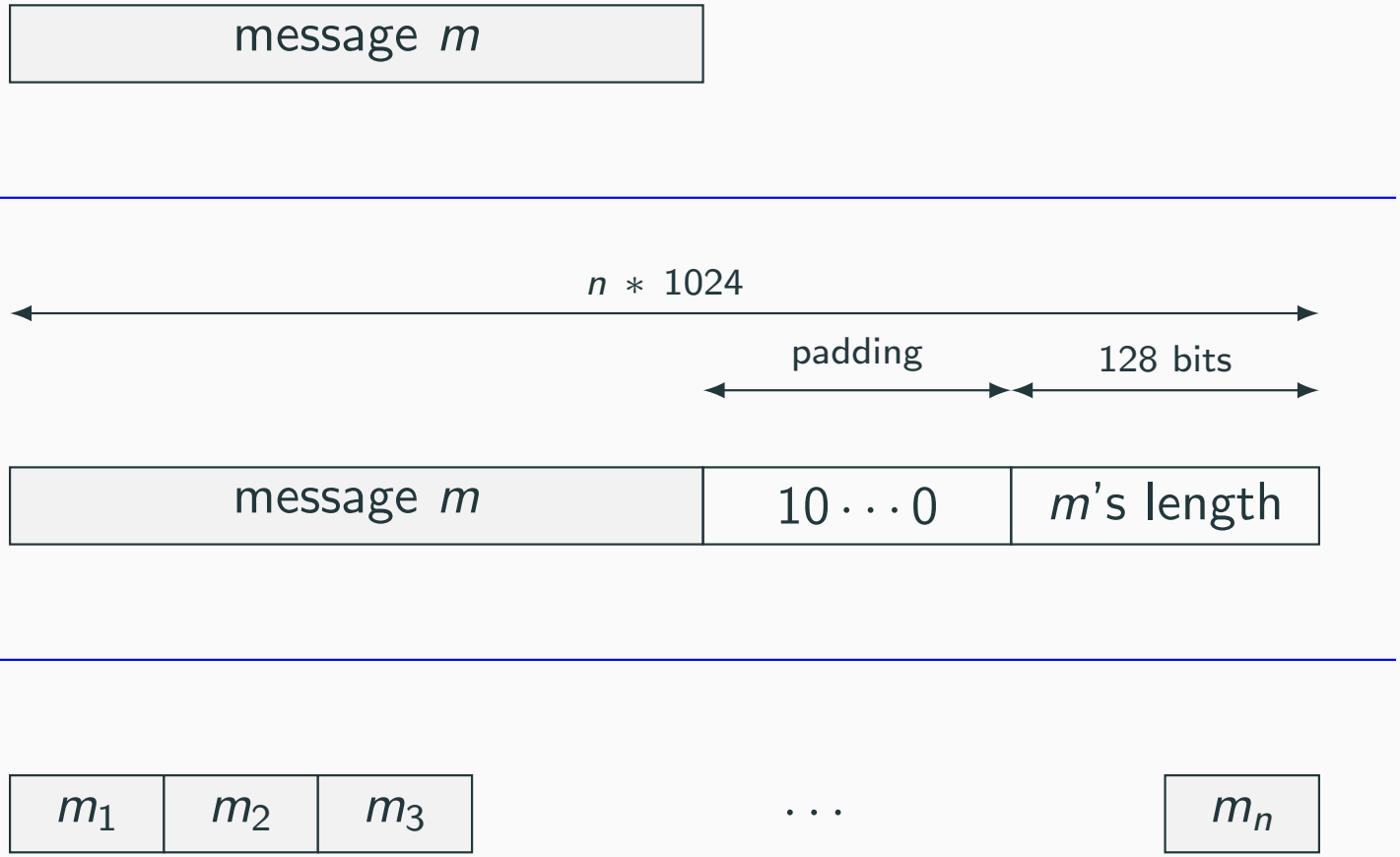
- MD4 – developed by Rivest in 1990. It was the starting point for the development of a series of similar hash functions;
- SHA (Secure Hash Algorithm) or SHA-0 – developed by NSA in 1993 (withdrawn shortly after publication because of some flaw);
- MD5 – the strengthened successor of MD4 (Rivest 1995);
- SHA-1 – developed by NSA in 1995; not longer approved after 2010;
- SHA-2 family includes 6 hash functions, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 (the last two are truncated versions of SHA-512).

Case study: the SHA-2 family

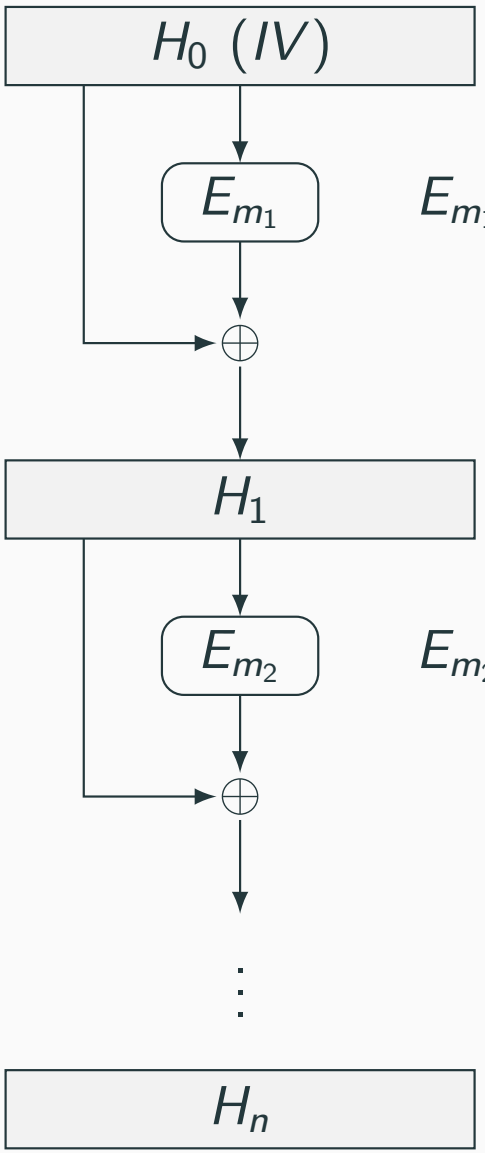
A bit of history:

- SHA-2 (Secure Hash Algorithm 2) was first published in 2001 by NSA. It includes 6 hash functions (see previous slide);
- SHA-2 is built using the MD transform and the Davies–Meyer compression function;
- All six functions in SHA-2 have the same structure, differing only in the number of rounds, shift values, and additive constants.

SHA 512: message padding



SHA 512



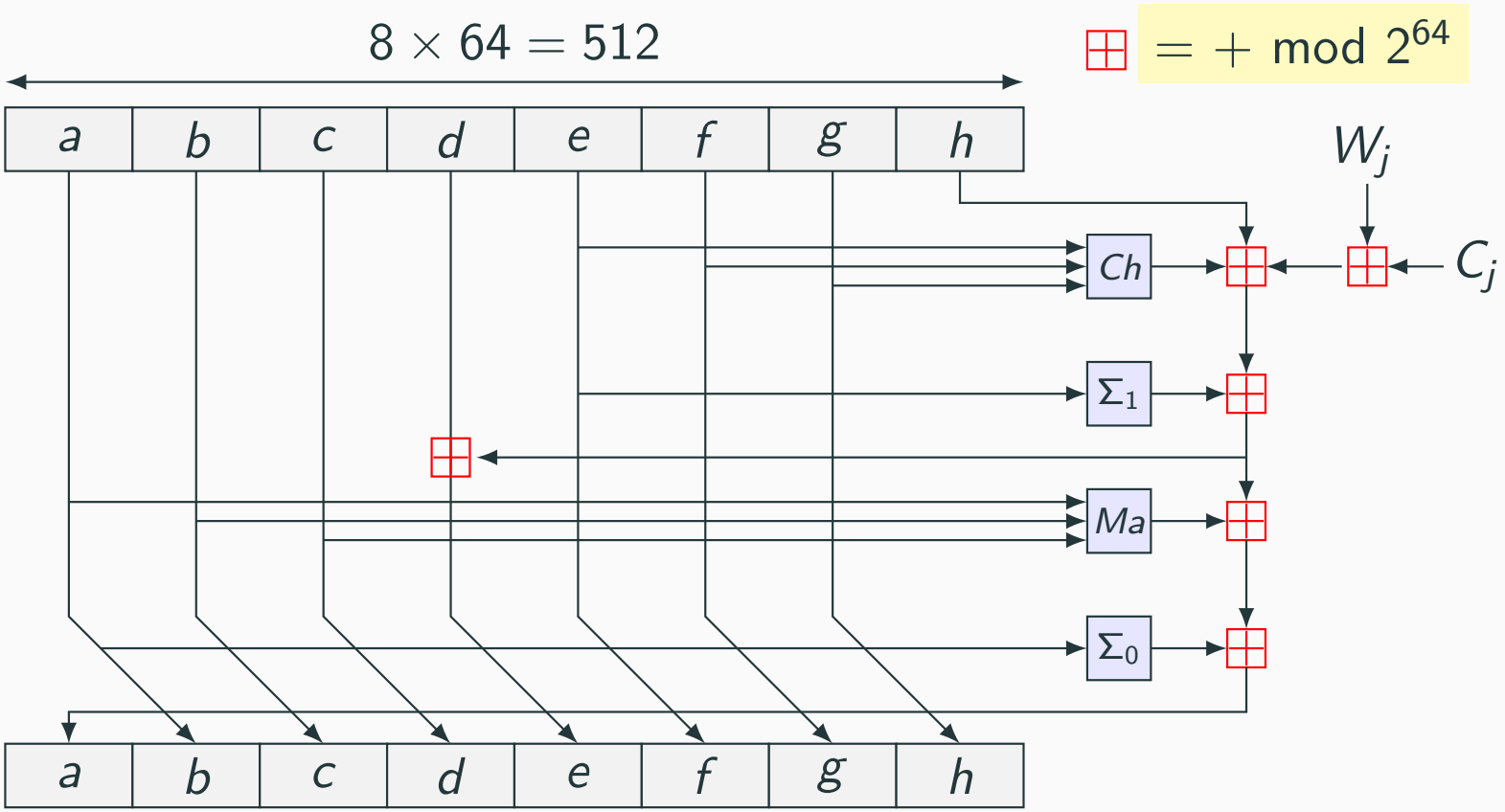
512-bit initialize vector

E_{m_1} works in 80 rounds

E_{m_2} works in 80 rounds

512-bit message digest

SHA 512: the encryption E



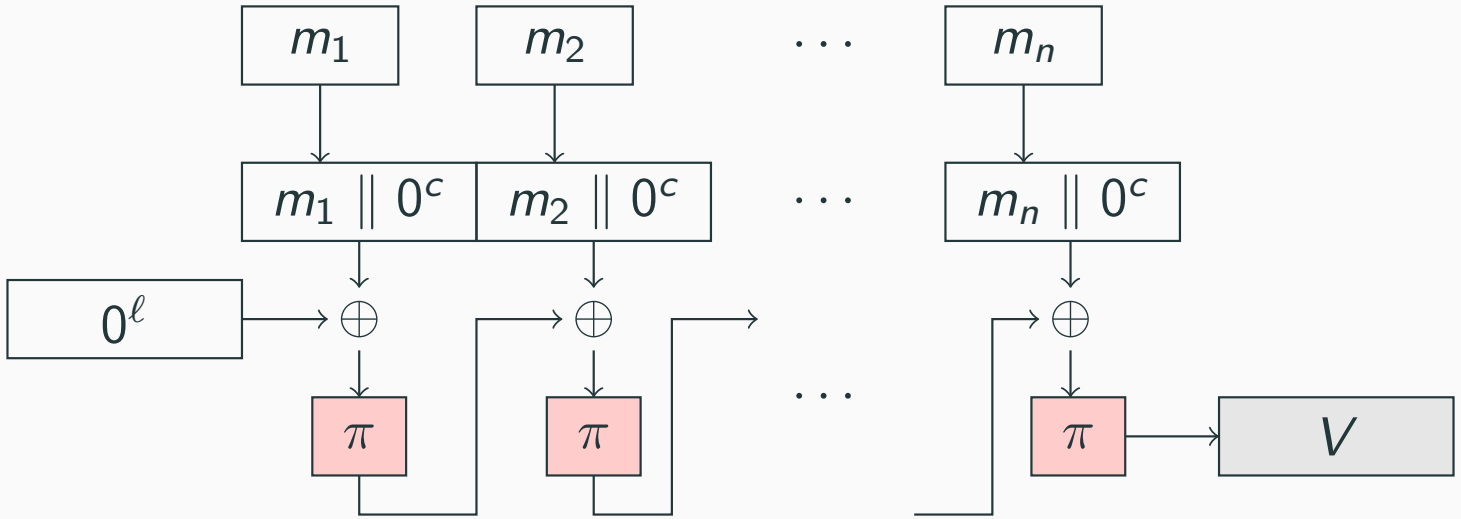
$Ch(e, f, g) = (e \wedge f) \oplus (\neg e \wedge g)$, $Ma(a, b, c) = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$

$\Sigma_0(a) = (a \ggg 28) \oplus (a \ggg 34) \oplus (a \ggg 39)$
$\Sigma_1(e) = (e \ggg 14) \oplus (e \ggg 18) \oplus (e \ggg 41)$

W_j are computed from the message block, C_j are given constants

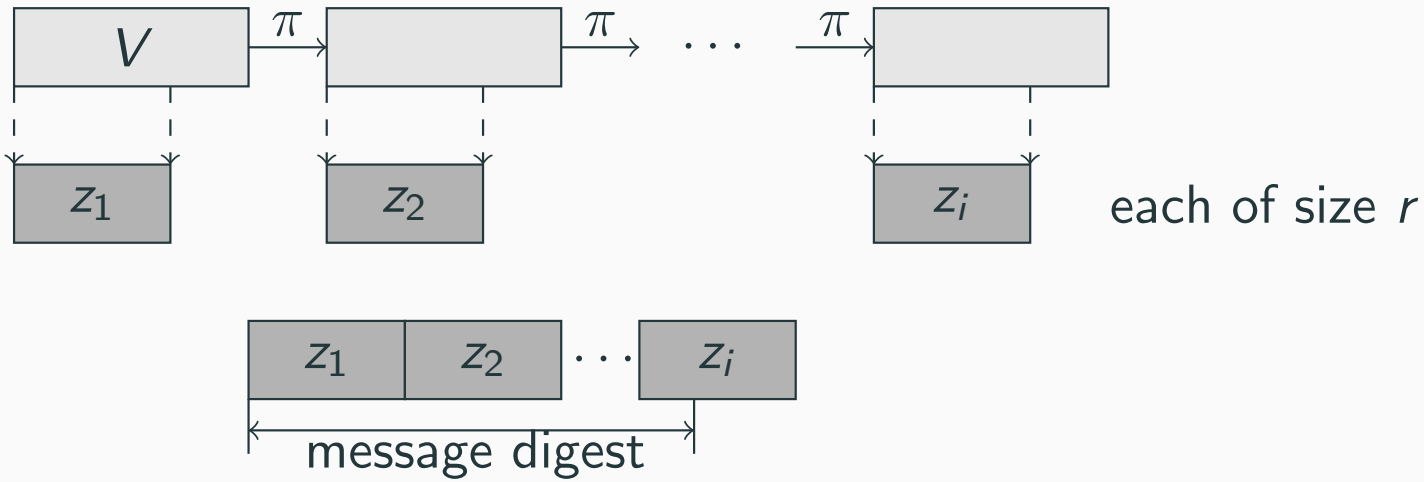
The sponge construction

- Choose a permutation $\pi : \{0,1\}^\ell \to \{0,1\}^\ell$ (π has no key!) and write $\ell = r + c$ (r is the rate and c is the capacity)
- Pad m by 10^*1 and divide it into r-bit blocks $m_1 \cdots m_n$ (padding is necessary even if the length of m is a multiple of r)
- Absorbing phase



The sponge construction

- Squeezing phase



Theorem 15

If π is a random permutation and 2^{ℓ} and 2^{c} are super-poly, then the sponge construction yields a CRHF.

The sponge construction is the basis of SHA-3 standard.

Case study: SHA-3

The sponge construction is the basis of SHA-3 standard:

1. SHA3-224: $\ell = 1600$, $r = 1152$, $c = 448$
2. SHA3-256: $\ell = 1600$, $r = 1088$, $c = 512$
3. SHA3-384: $\ell = 1600$, $r = 832$, $c = 762$
4. SHA3-512: $\ell = 1600$, $r = 576$, $c = 1024$

In all cases,

- $\pi = \mathit{Keccak\text{-}f}[1600]$
- The suffix 01 is appended to the message before processing. The suffix supports domain separation, i.e., it distinguishes the inputs to Keccak arising from different SHA-3 functions.

Reading and exercise guide

Course readings:

1. Pages 167-202 from [1].

I recommend that you read the SHA-2 and SHA-3 standards from NIST:

- `https://csrc.nist.gov/Projects/Hash-Functions`

References

[1] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

[2] Winternitz, R. S. (1984). A secure one-way hash function built from des. In *1984 IEEE Symposium on Security and Privacy*, pages 88–88.

Message Authentication Codes

Prof.dr. Ferucio Laurențiu Țiplea

Spring 2023

Department of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași 700506, Romania

e-mail: ferucio.tiplea@uaic.ro

Outline

Message authentication codes

Construction of MAC schemes

- CBC-MAC
- CMAC
- HMAC

Authenticated encryption

Message authentication codes

Message integrity

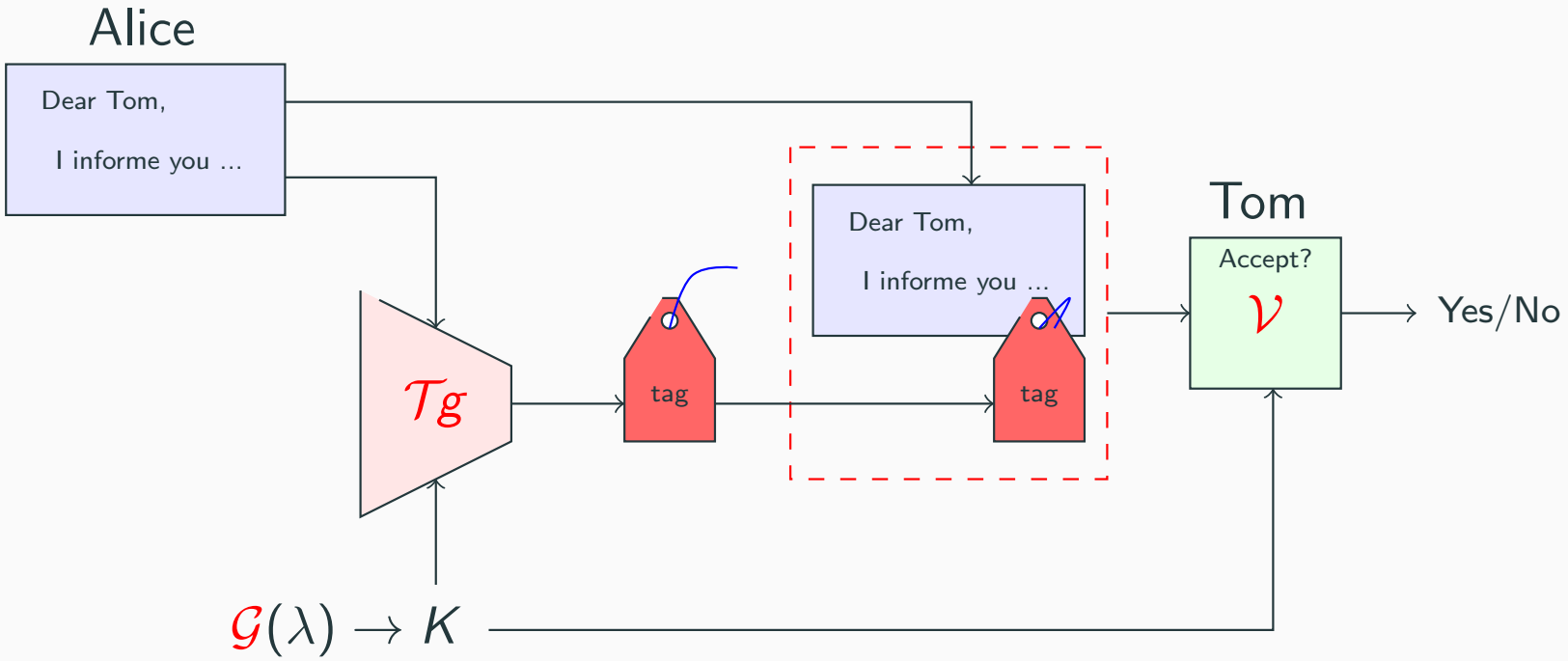
Message integrity (MI) = a very important cryptographic property

1. MI is not implied by confidentiality;
   - 1.1 In MI, the message need not be a secret;
   - 1.2 Encryption schemes such as stream ciphers do not guarantee MI;
   - 1.3 Changing an encrypted field in a ciphertext might result in valid plaintext by decryption and so, MI cannot be checked;
2. MI is not possible without a secret key, except for cases where MI is used to detect random errors in communication.

For historical reasons, message authentication is usually used instead of message integrity.

Message authentication codes: pictorial view

Message authentication codes (MACs) = used to prove message integrity based on a shared secret key between parties



$\mathcal{S} = (\mathcal{G}, \mathcal{T}g, \mathcal{V})$ – MAC system

Message authentication codes: formal definition

Definition 1

A MAC system over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ is a triple of algorithms $\mathcal{S} = (\mathcal{G}, \mathcal{T}g, \mathcal{V})$ such that:

1. $\mathcal{G}$ is the key generator;
2. $\mathcal{T}g$ is a PPT algorithm, called the tag generation algorithm, which outputs a tag $t \in \mathcal{T}$ when invoked on a key $K \in \mathcal{K}$ and a message $m \in \mathcal{M}$;
3. $\mathcal{V}$ is a DPT algorithm, called the verification algorithm, which outputs accept (or 1) or reject (or 0) when invoked on a key K, a message m, and a tag t.

Soundness: $P(\mathcal{V}(K, m, \mathcal{T}g(K, m)) = \texttt{accept}) = 1$

MAC security game

Experiment $MAC_{\mathcal{A},\mathcal{S}}(\lambda)$

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$;
2: The adversary $\mathcal{A}$ queries the challenger several times (sends messages and receive tags);
3: Eventually, $\mathcal{A}$ outputs a candidate forgery pair (m, t) not among the pairs it has.

The advantage of $\mathcal{A}$ is $Adv^{mac\text{-}forge}_{\mathcal{A},\mathcal{S}}(\lambda) = P(MAC_{\mathcal{A},\mathcal{S}}(\lambda))$, that is the probability that $\mathcal{A}$ wins the above game (i.e., t is a valid tag for m).

Definition 2

$\mathcal{S}$ is a secure MAC system if $Adv^{mac\text{-}forge}_{\mathcal{A},\mathcal{S}}(\lambda)$ is negligible, for all PPT adversaries $\mathcal{A}$.

Construction of MAC schemes

Construction of MAC schemes

1. MAC schemes from PRF
2. MAC schemes from hash functions
3. MAC schemes from PRF and hash functions

CBC-MAC

Cipher Block Chaining Message Authentication Code (CBC-MAC) is a MAC algorithm based on the CBC operation mode of a block cipher.

CBC-MAC is one of the oldest and most popular MAC scheme.

A brief history of CBC-MAC:

1. The idea of using block ciphers for data authentication was first described by Campbell in 1978 [8, 9];

2. Appendix F in FIPS PUB 81 [12] describes data authentication using DES in the CBC and CFB operation modes;

CBC-MAC

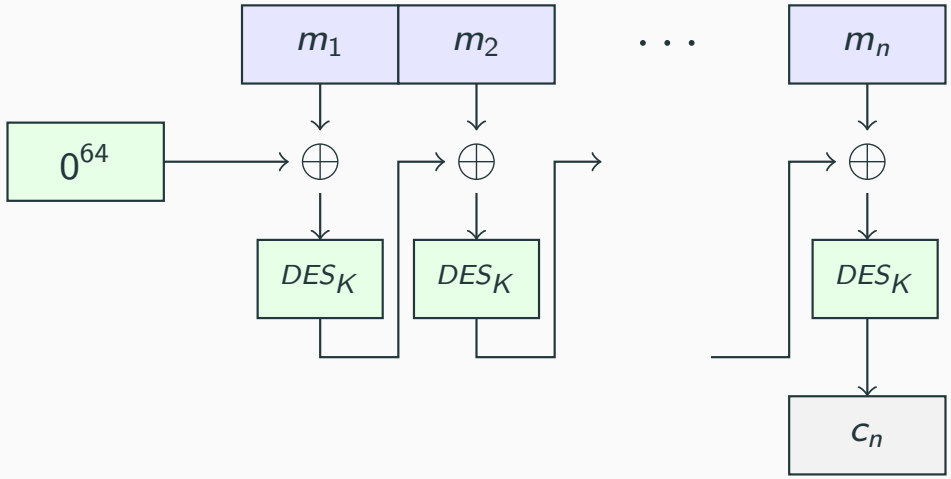
A brief history of CBC-MAC (cont.):

3. ANSI X9.9 (1982, 1986) [2, 3] and FIPS PUB 113 (1985) [11] define the first CBC-MAC standards;

4. ANSI X9.19 (1986) [4] defines a strengthened version of ANSI X9.9 MAC, called Retail MAC;

5. ISO 8731-1 (1987) [13], ISO 8731-2 (1987) [14], and ISO/IEC 9797 (1989) [15] specify variants of CBC-MAC using DES.

CBC-MAC



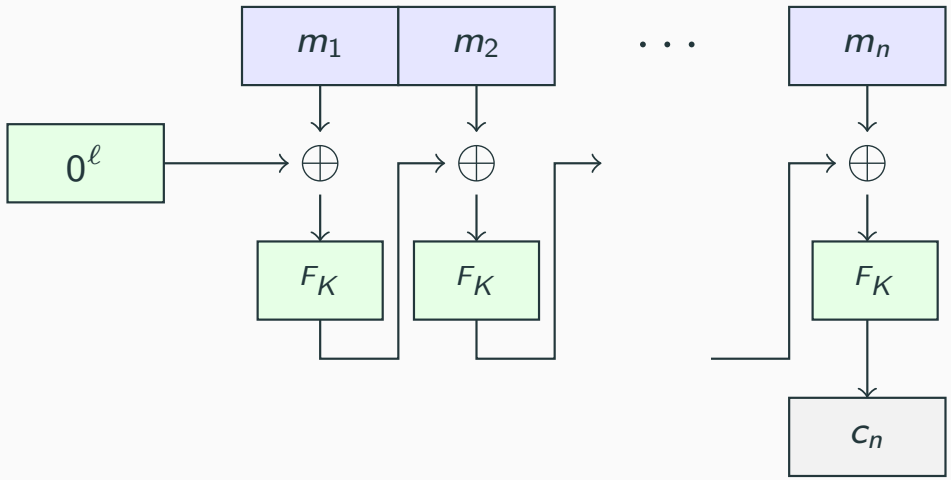
1. MAC: the tag is the leftmost 32, 48, or 64 bits of c_n;
2. Retail MAC: the tag is computed from $DES_K(DES_{K'}^{-1}(c_n))$, where K' is a new key.

[21] showed that the ANSI X9.19 retail MAC based on an n-bit block cipher offers no increased strength against exhaustive key search if about $2^{n/2}$ known text-MAC pairs are available.

CBC-MAC security

The first security proof for CBC-MAC is that from [5], under the following constraints:

1. the cipher is a secure PRF;
2. the class of messages consists of all messages with the same number of blocks of length ℓ, where ℓ is the block length of the PRF;
3. the tag is the last cipher block.



CBC-MAC security

A second security proof [7] relaxes the second constraint above, asking that the class of messages is the set of all messages whose number of blocks is at most n, where n is polynomial bounded, and no message is a prefix in another. This may be called prefix-free security.

Theorem 3

If F is a secure PRF, then CBC-MAC(F) achieves prefix-free security.

We may say that the CBC-MAC(F) construction gives rise to a new PRF that is prefix-free secure, provided that F is secure.

CBC-MAC(F) is a block-wise function, while F is a bit-wise function!

CBC-MAC: the good and the bad

CBC-MAC(F) is secure when used with messages that all have the same number of blocks! [5]

Drawback: all messages for which the MAC is built must have one fixed length, and that length must be a multiple of the block size!

If MAC is built for messages with variable number of blocks then:

1. Let $t = CBC\text{-}MAC(m_1 m_2)$ (a message with two blocks);
2. Then, $CBC\text{-}MAC(m_1 m_2 (m_1 \oplus t) m_2) = t$!

Remark that this works because $m_1 m_2$ is a prefix in $m_1 m_2 (m_1 \oplus t) m_2)$.

How do we process messages that cannot exactly be split into blocks of a certain length?

From CBC-MAC to CMAC

1. Encrypted MAC (EMAC): as CBC-MAC but one more iteration is added at the end, with another key. The idea first appeared in Retail MAC and then in [1];
2. XCBC (the three-key construction) [6]: uses two more keys, one to randomize the last block when it does not need padding, and the other one to randomize the the last block when it needs padding;
3. One-key MAC (OMAC) [16, 17, 18, 19]: replaces the two supplementary keys in XCBC by two values computed from the input data;
4. Cipher-based MAC (CMAC) [10]: the two supplementary keys in XCBC are calculated slightly differently but equivalent to the OMAC-1 method (a variation of OMAC).

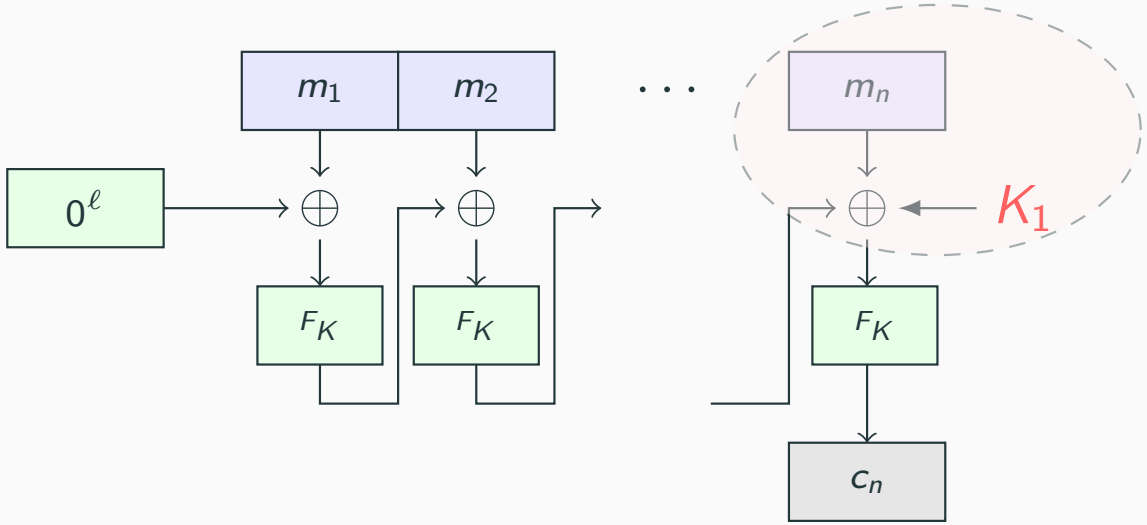
CMAC is the best approach to building a CBC-MAC!

CMAC: overview

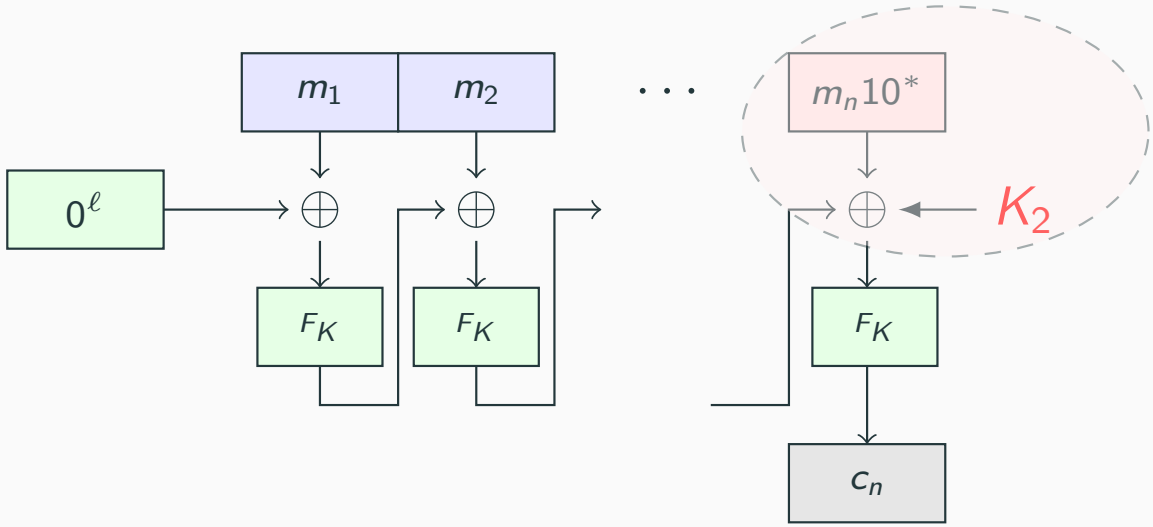
1. CMAC depends on the choice of an underlying symmetric key block cipher;
2. Consider bellow a PRF $F : \mathcal{K} \times \{0,1\}^{\ell} \to \{0,1\}^{\ell}$, where the keys have size k;
3. The message for which a tag is to be computed is split into ℓ-bit blocks. If the last block is incomplete, it is padded by 10^*;
4. Then, the last message block is "randomized" and F_K is applied in the CBC operation mode to the entire message. The randomization acts as a prefix-free encoding.

CMAC: tag generation

All message blocks have size ℓ



All message blocks have size ℓ, except for the last one



CMAC: subkey generation

1. $L := F_K(0^\ell)$;
2. If $L(1) = 0$, then $K_1 := L \ll 1$; else, $K_1 := (L \ll 1) \oplus R_\ell$;
3. If $K_1(1) = 0$, then $K_2 := K_1 \ll 1$; else, $K_2 := (K_1 \ll 1) \oplus R_\ell$;
4. "$X \ll 1$" means to discard the leftmost bit of X and to add 0 on the right;
5. $R_\ell = c_{\ell-1} \cdots c_1 c_0$, where $P(x) = x^\ell + c_{\ell-1}x^{\ell-1} + \cdots + c_1 x + c_0$ is the first (in lexicographic order) irreducible polynomial of degree ℓ with the minimum possible number of nonzero terms.

CMAC security

Remark that the CMAC construction processes the (last block of the) message so that with overwhelming probability, no message is a prefix of another one.

Definition 4

A randomized prefix-free encoding on a set of messages is a keyed function f with the following properties:

1. the output of f is always a sequence of blocks of the same length ℓ;
2. $P(f(K, m_0) \preceq_{prefix} f(K, m_1))$ is negligible, for any two distinct messages m_0 and m_1 (the probability is computed over the random choice of K).

Example 5

If $m = m_1 \cdots m_n \in (\{0,1\}^\ell)^n$, then $f(K, m) = m_1 \cdots m_{n-1}(m_n \oplus K)$ is a randomized prefix-free encoding.

CMAC security

Are the encodings below randomized prefix-free encodings?

1. $(m_1, \ldots, m_k) \rightarrow (\langle k \rangle, m_1, \ldots, m_k)$;
2. $(m_1, \ldots, m_k) \rightarrow (m_1 \| 0, \ldots, m_{k-1} \| 0, m_k \| 1)$, where m_i have length $\ell - 1$.

Let $CMAC(F, f)$ demote the CMAC scheme built on a PRF F and a randomized prefix-free encoding f.

Theorem 6

If F is a prefix-free secure PRF and f is a randomized prefix-free encoding, then CMAC(F,f) is a secure MAC scheme.

MACs from UHFs

Let H be a keyed hash function that returns ℓ-bit message digests and F be a PRF on ℓ-bit message blocks. Define the following family of functions:

$$F_H((K_1, K_2), m) = F(K_2, H(K_1, m))$$

Theorem 7

If H is a UHF and F is a PRF, then F_H is a PRF.

F_H can be used to define a MAC scheme for arbitrary large messages.

MACs from CRHFs: HMAC

Let H be a (keyless) hash function defined by the MD transform from a compression function $h(K, m)$. Define F_H by

$$F_H((K_1, K_2), m) = H(K_2 \parallel H(K_1 \parallel m))$$

Remark that K_1 and K_2 are treated as the first message block in H!

Theorem 8

If h and h' given by $h'(K, m) = h(m, K)$ are PRFs, then F_H is a PRF.

The HMAC construction uses one single key K from which two keys are derived: $K_1 = K \oplus ipad$ and $K_2 = K \oplus opad$.

HMAC-SHA1 and HMAC-SHA256 are instances of the above construction, with $H = SHA1$ and $H = SHA256$.

Authenticated encryption

Ciphertext integrity

Consider the ciphertext integrity experiment between a cipher $\mathcal{S}$ and an adversary $\mathcal{A}$ to check the ability of $\mathcal{A}$ to construct valid ciphertexts:

Experiment $CI_{\mathcal{A},\mathcal{S}}(\lambda)$

1: The challenger generates a key $K \leftarrow \mathcal{G}(\lambda)$
2: The adversary $\mathcal{A}$ queries the encryption oracle
3: Eventually $\mathcal{A}$ outputs a candidate ciphertext c different than the ones obtained by queries
4: If c is a valid ciphertext then return 1, else return 0.

The advantage of $\mathcal{A}$ is $Adv^{ci}_{\mathcal{A},\mathcal{S}}(\lambda) = P(CI^{ci}_{\mathcal{A},\mathcal{H}}(\lambda) = 1)$

Authenticated encryption

Definition 9

A cipher $\mathcal{S}$ provides ciphertext integrity (CI) if $Adv^{ci}_{\mathcal{A},\mathcal{S}}(\lambda)$ is negligible for all PPT algorithms $\mathcal{A}$.

Definition 10

A cipher $\mathcal{S}$ provides authenticated encryption (AE) if:

1. $\mathcal{S}$ is IND-CPA secure
2. $\mathcal{S}$ provides CI.

Theorem 11

If $\mathcal{S}$ is AE secure then it is IND-CCA secure.

Constructing AE secure ciphers

One popular way to construct AE secure ciphers is to combine an IND-CPA secure cipher with a secure MAC. There are two main variants:

1. Encrypt-then-MAC (EtM)
   - 1.1 $c \leftarrow \mathcal{E}(K, m)$, $t \leftarrow \mathcal{T}g(K', c)$, output (c, t)
   - 1.2 Used in IPsec, TLS 1.2 and later versions, and in the NIST standard GCM
2. MAC-then-Encrypt (MtE)
   - 2.1 $t \leftarrow \mathcal{T}g(K', m)$, $c \leftarrow \mathcal{E}(K, (m, t))$, output c
   - 2.2 Used in SSL 3.0, TLS 1.0, and in 802.11i WiFi encryption protocol

The keys K and K' must be chosen independently!

Encrypt-then-MAC

Theorem 12

If $\mathcal{S}$ is an IND-CPA secure cipher and $\mathcal{S}'$ is a secure MAC, then the EtM construction is AE secure.

Common mistakes in implementing the EtM construction:

1. Use the same key for the cipher and the MAC;
2. Apply the MAC only to part of the ciphertext (we may loose ciphertext integrity) – discovered in 2013 at `RNCryptor` facility in Apple's iOS.

MAC-then-Encrypt

MtE is not generally secure:

1. The attack POODLE on SSL 3.0
2. Padding oracle timing attack in TLS 1.0
3. Informative error messages in TLS 1.0

There are secure instances of MtE:

1. The randomized counter mode of the cipher assures AE security

Reading and exercise guide

Course readings:

1. Pages 105-140 and 172-177 from [20].

References

[1] (1995). *Integrity Primitives for Secure Information Systems: Final Ripe Report of Race Integrity Primitives Evaluation*. Springer-Verlag, Berlin, Heidelberg.

[2] American National Standard Insitute (1982). Financial institution message authentication (wholesale). Technical Report ANSI X9.9, American Bankers Association.

[3] American National Standard Insitute (1986a). Financial institution message authentication (wholesale). Technical Report ANSI X9.9 (standard revision), American Bankers Association.

[4] American National Standard Insitute (1986b). Financial institution retail message authentication. Technical Report ANSI X9.19, American Bankers Association.

[5] Bellare, M., Kilian, J., and Rogaway, P. (2000). The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61:362–399.

[6] Black, J. and Rogaway, P. (2000). Cbc macs for arbitrary-length messages: The three-key constructions. In Bellare, M., editor, *Advances in Cryptology — CRYPTO 2000*, pages 197–215, Berlin, Heidelberg. Springer Berlin Heidelberg.

[7] Boneh, D. and Shoup, V. (2020). A graduate course in applied cryptography.

[8] Campbell, C. (1978a). Design and specification of cryptographic capabilities. In Branstad, D. K., editor, *Conference on Computer Security and the Data Eneryption Standard (Feb 1977)*, volume 500-27 of *NBS Special Publication*, pages 54–66. U.S. Department of Commerce, National Bureau of Standards.

[9] Campbell, C. (1978b). Design and specification of cryptographic capabilities. *IEEE Communications Society Magazine*, 16(6):15–19.

[10] Dworkin, M. (2005). Recommendation for block cipher modes of operation: The CMAC mode for authentication. Technical Report NIST Special Publication 800-38B, U.S. Department of Commerce, Washington, D.C.

[11] Federal Information Processing Standard Publication 113 (1985). Computer data authentication. Technical Report FIPS PUB 113, National Bureau of Standards.

[12] Federal Information Processing Standard Publication 81 (1980). Des modes of operation. Technical Report FIPS PUB 81, National Bureau of Standards.

[13] International Organization for Standardization (1987a). Banking – approved algorithms for message authentication. part i: Dea. Technical Report ISO 8731-1.

[14] International Organization for Standardization (1987b). Banking – approved algorithms for message authentication. part ii: Message authenticator algorithms. Technical Report ISO 8731-2.

[15] International Organization for Standardization (1989). Data cryptographic techniques ? data integrity mechanism using a cryptographic check function employing a block cipher algorithm. Technical Report ISO/IEC 9797.

[16] Iwata, T. and Kurosawa, K. (2002). OMAC: One-key CBC MAC.

[17] Iwata, T. and Kurosawa, K. (2003a). OMAC: One-key CBC MAC. In *Pre-proceedings of Fast Software Encryption, FSE 2003*, pages 129–153. Springer-Verlag.

[18] Iwata, T. and Kurosawa, K. (2003b). OMAC: One-key CBC MAC – addendum.

[19] Iwata, T. and Kurosawa, K. (2003c). Stronger security bounds for OMAC, TMAC, and XCBC. In Johansson, T. and Maitra, S., editors, *Progress in Cryptology - INDOCRYPT 2003*, pages 402–415, Berlin, Heidelberg. Springer Berlin Heidelberg.

[20] Katz, J. and Lindell, Y. (2021). *Introduction to Modern Cryptography*. CRC Press, New York, 3rd edition.

[21] Preneel, B. and van Oorschot, P. (1996). Key recovery attack on ANSI X9.19 retail MAC. *Electronics Letters*, 32:1568–1569(1).