Quantum Cryptography notebook

Eva Ricci, Giacomo Bertelli

Politecnico di Torino,
Master Degree in Quantum Engineering

February 13, 2025

These notes are based on the lectures of
Quantum Cryptography by Professor Adami but they have not
been revised by the teacher and may thus contain
mistakes.
If you find any please let us know.

If you found these notes useful, you can offer us a coffee at the link below:

https://paypal.me/deskclam°

Contents

1. Intro and Caesar cipher

- Key space: set of all possible keys
- Size of the key space $K = \#K$ must be large enough to avoid brute force

1.1. Kerchoff rules

1. The system must be indecipherable
2. A cryptographic system must be secure even if the attacker knows all the details about the system with the exception of the key
3. It must be possible to communicate without written notes
4. It must apply to the telegraph
5. It must be portable
6. It must be simple

1.2. Caesar's cipher

We translate letters to numbers $A \to 0, B \to 1, \dots$ and the key is given by how 'A' is encrypted. For example:

$$\begin{gathered} A \to S \equiv 0 \to 18 \\ \Downarrow \\ e(0) = 0 + 18 \quad (18 \text{ is the key}) \end{gathered} \tag{1.1}$$

$e(x)$ is the remainder of the division

$$\frac{x + 18}{26} \forall x \in [0, 25] \tag{1.2}$$

2. Modular algebra

2.1. Definition

$$\begin{gathered} a, r, m \in \mathbb{Z} \\ m > 0 \\ a \equiv r \bmod m \quad \text{if } m \text{ divides } (a - r) \\ \text{or in symbols: } m \mid a - r \end{gathered} \tag{2.1}$$

2.2. Properties

2.2.1. Sum

$$\begin{cases} a = r \bmod m \\ b = v \bmod m \end{cases} \implies a + b = (r + v) \bmod m \tag{2.2}$$

Proof:

$$\begin{gathered} a \equiv r \bmod m \Rightarrow a = mk + r \\ b \equiv v \bmod m \Rightarrow b = mh + b \\ \Downarrow \\ a + b = m(k + h) + r + v \end{gathered} \tag{2.3}$$

This means that m divides $(a + b) - (r + v)$, which is to say that

$$(a + b) \equiv (r + v) \bmod m \tag{2.4}$$

□

2.3. Product

$$\begin{cases} a = r \bmod m \\ b = v \bmod m \end{cases} \implies a \cdot b = (r \cdot v) \bmod m \tag{2.5}$$

Proof:

$$\begin{aligned} ab &= m^2kh + mkv + mhr + rv \\ &= m(mkh + kv + hr) + rv \end{aligned} \tag{2.6}$$

□

2.4. Taking factors

Pay attention when taking factors in modular algebra.

Example:

$$a \cdot b \bmod p \equiv (a \bmod p \cdot b \bmod p) \bmod p \tag{2.7}$$

Example:

$$\begin{aligned}(x - x^2) \bmod p &\equiv [x(1-x)] \bmod p \\ &\equiv (x \bmod p) \cdot [(1-x) \bmod p] \bmod p\end{aligned} \tag{2.8}$$

2.5. Inverse in $\mathbb{Z}_m$

The inverse of a in $\mathbb{Z}_m$ is a number b s.t. $ba = ab = 1 \bmod m$

Theorem 2.1 (Uniqueness of the inverse): The inverse is unique in $\mathbb{Z}_m$

Proof: Suppose $\exists b, b' \in \mathbb{Z}_m$ s.t. $ba = b'a = 1 \bmod m$. Then

$$\begin{gathered}ba \equiv 1 \bmod m \Rightarrow ba = 1 + mk \text{ for some } k \in \mathbb{Z} \\ b'a \equiv 1 \bmod m \Rightarrow b'a = 1 + mk' \text{ for some } k' \in \mathbb{Z} \\ \Downarrow \\ (b - b')a = m(k - k') \equiv 0 \bmod m \\ \Longrightarrow b = b' \bmod m\end{gathered} \tag{2.9}$$

proving that the inverse is unique (if it exists). □

Theorem 2.2 (Existence of the inverse 1):

$$\text{If } a \text{ and } m \text{ are coprime} \Longrightarrow \exists\ a^{-1} \in \mathbb{Z}_m \tag{2.10}$$

Proof: We solve $ax = 1 \bmod m \iff x = 1 + mk$ for some k, which is to say that $a | 1 + mk$.

We should consider a watch with a hours and find a k for which $a = 1 + mk$:

1. If we try $k = 0$ we can see that it doesn't work
2. If we try $k = 1$ and we move m steps, when $m > a$ we wrap around the clock and get an effective increment on the watch equal to n which is the remainder of m/a:

$$m \equiv n \bmod a \tag{2.11}$$

We can observe that $\gcd(n, a) = 1$ because otherwise we would get that exists s which is a common factor, that divides both n and a, in formulas: $n = n' \cdot s$ and $a = a' \cdot s$. For Equation (2.11) we could thus write that, for some $h \in \mathbb{Z}$:

$$\begin{aligned}m &= n + ah \\ &= n's + a'sh \\ &= s(n' + a'h) \Longrightarrow s \mid m\end{aligned} \tag{2.12}$$

which means that m and a are not coprime (since s divides both of them), contradicting the hypothesis (Equation (2.10)).

We can then notice that the motion of the point on the watch is periodic since the number of position is finite (so they will repeat at some point) and since, given a position, the next one is fixed.

We can also notice that the period cannot be $< a$ because, otherwise if $T < a$ then

$$\underbrace{Tn}_{<an} = \tau a \quad T, \tau \in \mathbb{N} \tag{2.13}$$

Since $Tn = \tau a$ is a common multiplier of both n and a, we get that $\text{lcm}(a, n) < an$ (since $Tn < an$) which means that a and n are not coprime, so the period is a. This means that:

1. The point visits all possible sites, including a
2. $\exists \bar{k}$ s.t. $a = 1 + \bar{k}m \implies \exists a^{-1}$

□

Theorem 2.3 (Existence of the inverse 2):

$$\text{If } a \text{ and } m \text{ are NOT coprime} \implies \nexists\ a^{-1} \in \mathbb{Z}_m \tag{2.14}$$

Proof: Having a^{-1} means that exists b s.t. $ba = 1 + mk$ for some k.

Let c be a common non-trivial divisor of a and m, then $c \mid ba$ while $c \nmid 1 + mk$, so $b = a^{-1}$ does not exist.

□

2.6. Domain of integrity

Definition 2.1 (Domain of integrity): A **domain of integrity** is a ring (see Section 3.5. [p. 9]) where product is commutative and with the following property: $a \cdot b = 0 \implies a = 0 \vee b = 0$

In regular algebra:

$$\alpha\beta = 0 \implies \text{either } \alpha = 0 \text{ or } \beta = 0 \tag{2.15}$$

In modular algebra:

$$\begin{gathered} \alpha\beta \equiv 0 \bmod p, p \text{ prime} \\ \Updownarrow \\ \alpha\beta = kp \\ \text{for some } k \in \mathbb{Z} \end{gathered} \tag{2.16}$$

Theorem 2.4:

If p is not prime then $\mathbb{Z}_p$ is not a domain of integrity.

If p is prime then $\mathbb{Z}_p$ is a domain of integrity.

2.7. Euler's Φ

Definition 2.2 (Euler's Φ):

$$\begin{gathered} n \in \mathbb{N} \\ \text{Given } \Phi(n) = \#\{p < n\} + 1, p \text{ coprime with } n \end{gathered} \tag{2.17}$$

Theorem 2.5 (Euler's-Fermat):

$$(a, m) \text{ coprime} \implies a^{\Phi(m)} \equiv 1 \bmod m \tag{2.18}$$

Corollary 2.1: If $\gcd(a, m) = 1$ then $\forall b$ we have that

$$a^b \equiv a^{b'}, \quad b \equiv b' \bmod \Phi(m) \tag{2.19}$$

2.8. Exercises on modular algebra

Exercise 2.1: Find all x s.t. $x \equiv x^s \bmod p$.

$$\begin{aligned} x - x^2 &\equiv 0 \bmod p \\ x(1 - x) &\equiv 0 \bmod p \end{aligned} \tag{2.20}$$

In $\mathbb{R}$ this would imply that either $x = 0$ or $x = 1$ since $\mathbb{R}$ is a domain of integrity.

***p* prime:**

Since for Theorem 2.4 [p. 4] $\mathbb{Z}_p$ with p prime is a domain of integrity as well we can conclude that the solutions are $x = 0$ and $x = 1$.

***p* not prime:**

Let's consider the example $p = 15$

2.9. Vigenère and affine ciphers

2.9.1. Vigenère

In the Vigenère cipher every letter is encoded according to a different permutation, according to a permutation table and the encryption key: the table is reported in Table 1 [p. 6].

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

Table 1: Vigenère table of permutations (tabula recta)

Example:

1. **plaintext**: "*It was late evening*"
2. **key**: "*KAFKA*"
3. **message**: "*St bks vayo efessnq*"

2.9.2. Affine

In the affine cipher the plaintext is encrypted by mapping each letter in the correspondent numerical, the encryption is then done by using a simple mathematical expression and each number converted back in the correspondent letter. The expression is:

$$E(x) = (ax + b) \bmod m \tag{2.21}$$

where the x is the numerical correspondent of the letter of the plaintext to be encrypted and m is the length of the alphabet used. The relation for the decryption is:

$$D(x) = a^{-1}(x - b \bmod m) \tag{2.22}$$

3. Equivalence relations

Definition 3.1 (Equivalence relation): Let A be a set. A relation of equivalence is a law between couples of elements of A s.t. $x \sim y$.

3.1. Properties

1. Reflexivity:

$$x \sim x \tag{3.1}$$

2. Symmetry:

$$x \sim y \Longrightarrow y \sim x \tag{3.2}$$

3. Transitivity:

$$x \sim y \wedge y \sim z \Rightarrow x \sim z \tag{3.3}$$

3.2. Equivalence class

Definition 3.2 (Equivalence class): Let A be a set with an equivalence relation and let $x \in A$. The equivalence class $[x]$ of x is the subset of A with all the elements equivalent to x:

$$[x] = \{y \in A \text{ s.t. } y \sim x\} \tag{3.4}$$

Theorem 3.1: Two equivalence classes either coincide or are disjoint.

Proof: Suppose that $[x]$ and $[y]$ are not disjoint. Let us consider $z \in [x] \cap [y]$

Since $z \in [x]$ implies $z \sim x$ and $z \in [y]$ implies $z \sim y$ for symmetry and transitivity $\Longrightarrow x \sim y$.

Consider $u \in [x]$, then $u \sim x$ but since also $x \sim y$ we conclude by transitivity $u \sim y \Longrightarrow u \in [y] \Longrightarrow [x] \subset [y]$.

Analogously we obtain that $[y] \subset [x]$, thus we can conclude that $[x] = [y]$. □

3.3. Partitions

A partition of a set is a grouping of its elements into non-empty subsets, in such a way that every element is included in exactly one subset.

3.3.1. Properties

1. $$\forall E \in \mathcal{F}, E \neq \emptyset \tag{3.5}$$
2. $$\forall E, E' \in \mathcal{F}, E \neq E', E \cap E' = \emptyset \tag{3.6}$$

3.
$$\bigcup_{E\in\mathcal{F}}\{x\in E\}=A \tag{3.7}$$

Given A with "$\sim$", a family of subset of A is naturally induced:

$$\mathcal{F}=\{[x],x\in A\} \tag{3.8}$$

3.4. Relations

A relation in a set denotes a relationship between two objects in a set. A is a subset of $A\times A$.

In general a relation is not an equivalence of relation.

3.4.1. Equivalence classes

Example (Important): Given $a,b,r\in\mathbb{Z}$ we say:

$$a\sim b \tag{3.9}$$

if and only if exists $k\in\mathbb{Z}$ s.t. $a-b=kr$ (with r fixed).

1. Reflexivity:
$$a-a=0=0\cdot r\Longrightarrow a\sim a \tag{3.10}$$
2. Symmetry:
$$\begin{gathered}a-b=kr,\ b-a=-kr,\ -k\in\mathbb{Z}\\ \Downarrow\\ (a\sim b\Longrightarrow b\sim a)\end{gathered} \tag{3.11}$$
3. Transitivity:
$$\begin{gathered}a-b=kr\text{ and }b-c=hr\\ \Downarrow\\ a-c=(k+h)r\quad\text{with }(h+k)\in\mathbb{Z}\\ \Downarrow\\ a\sim b\text{ and }b\sim c\\ \Downarrow\\ a\sim c\end{gathered} \tag{3.12}$$

This is an equivalent relation: $a-b=kr\iff a=b\bmod r$.

Example: For $r=4$

$[0]=\{a\in\mathbb{Z}\text{ s.t. }a=4k\}=\{...,-8,-4,0,4,8,...\}$

$[1]=\{a\in\mathbb{Z}\text{ s.t. }a=4k+1\}=\{...,-7,-3,1,5,9,...\}$

$[2]=\{a\in\mathbb{Z}\text{ s.t. }a=4k+2\}=\{...,-6,-2,2,6,...\}$

$[3]=\{a\in\mathbb{Z}\text{ s.t. }a=4k+3\}=\{...,-5,-1,3,7,...\}$

$[4]=\{a\in\mathbb{Z}\text{ s.t. }a=4k+4\}=\{...,-8,-4,0,4,8,...\}=[0]$

The set $\mathbb{Z}$ is partitioned in 4 subsets called **modulo-rect classes**. This example can be used to introduce rings, since we have *sum* and *product*.

3.5. Rings

A set R is called a ring if it has 2 inner operations *sum* and *product* such that:

1. Sum is associative and commutative $\forall x, y, z \in \mathbb{R}$:

$$\begin{aligned} (x+y)+z &= x+(y+z) \\ x+y &= y+z \end{aligned} \tag{3.13}$$

2. $\exists$ a neutral element of the sum, called the "*zero*" "0" s.t.:

$$x+0=x \ \forall x \in \mathbb{R} \tag{3.14}$$

3. $\forall x \in \mathbb{R} \ \exists$ the *inverse* of x, x' with respect to the sum s.t.:

$$x+x'=0 \tag{3.15}$$

4. Product is associative $\forall x, y, z \in \mathbb{R}$:

$$(xy)z = x(yz) \tag{3.16}$$

5. Product is distributive with respect to the sum:

$$\begin{aligned} (x+y)z &= xz+yz \\ x(y+z) &= xy+xz \end{aligned} \tag{3.17}$$

We do not require that the product is commutative.

Remark: Modulo-rect classes $\mathbb{Z}_r = \{[0], ..., [r-1]\}$ form a ring which is commutative with respect to product ("commutative ring").

Given $\mathbb{Z}_r^*$, for some element there exists the inverse with respect to the product ($\exists [a]^{-1} \iff \gcd(a, r) = 1$)

4. Random number generators

4.1. True random number generators

Like coin tossing, noise in currents in semiconductors, measurements in chaotic systems (quantum measurements), for which the output is not reproducible.

4.2. Pseudo random number generators

The output is deterministic but the sequences generated using this type of generators have good statistical properties.

$$\begin{cases} s_0 \\ s_{i+1} = f(s_i, s_{i-1}, ..., s_{i-t}) \end{cases} \quad \text{t is called 'memory'} \tag{4.1}$$

4.2.1. Linear congruential generator

$$f(s) = as + b \bmod m \tag{4.2}$$

4.3. Cryptographically secure PRNG

A PRNG such that, even knowing a part of the stream $s_i, s_{i+1}, ..., s_{i+n}$ it is computationally infeasible to predict the following numbers of the stream.

5. Stream ciphers

5.1. Working principle

We encode the message we want to send into binary digits x_i. We have a key stream generator that generates a stream of s_i.

We transmit $y_i = x_i + s_i \bmod 2$.

Bob can retrieve x_i by doing $x_i = y_i + s_i \bmod 2$.

$\{s_0, s_1, ..., s_i\}$ is called *key stream.*

The sum modulo 2 is a good operation because it is *invertible* and it *"mixes"* well the digits.

5.2. Key stream based on PRNGs

Example:

$$\begin{cases} S_0 = \text{seed} \\ S_{i+1} = AS_i + B \bmod m \end{cases} \tag{5.1}$$

$m = (m_0, ..., m_{99})$ is public, A and B are 100-bits each and together they form the 200-bit key.

$|\mathcal{K}| = 2^{200}$

X_j are the words that Alice sends:

$X_1 = (x_0, x_1, ..., x_{99}), X_2 = (x_{100}, x_{101}, ..., x_{199})$

A possible attack

If Eve captures 300 bits X_1, X_2, X_3 then she can compute S_1, S_2, S_3 by subtraction, because $\forall i = 0, ..., 299, \quad X_i + S_i = Y_i$.

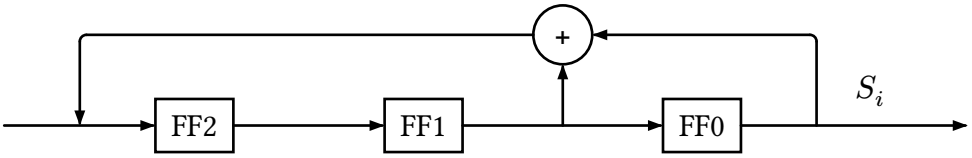
Now, $S_2 = AS_1 + B \bmod m$ and $S_3 = AS_2 + B \bmod m$ so:

$$S_2 - S_3 = A(S_1 - S_2) \bmod m \implies A = (S_2 - S_3)(S_1 - S_2)^{-1} \bmod m \tag{5.2}$$

supposing that $S_1 - S_2$ is coprime with m

$$B = S_2 + (S_3 - S_2)(S_1 - S_2)^{-1} S_1 \tag{5.3}$$

5.3. Linear shift feedback register (LFSR)

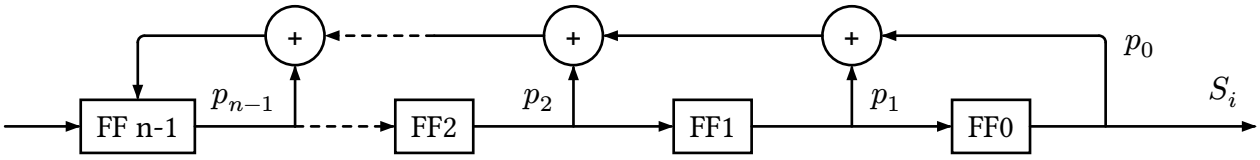


The initial state cannot be made of all zeroes and the sequence is periodic since the number of configuration is finite.

The maximal period is equal to $2^n - 1$ with n the number of registers.

A more generic scheme is drawn below, where the coefficients p_is represent if the circuit is closed or not:

- If $p_i = 1$ the correspondent connection is closed
- If $p_i = 0$ the correspondent connection is open



The initial state is given $(S_{n-1}, ..., S_1, S_0)$ and the following generic state is given by:

$$S_{n+i} = \sum_{j=0}^{n-1} p_j S_{i+j} \bmod 2 \tag{5.4}$$

If Eve knows:

- All the "y_is"
- The degree m
- A certain number of bits: $x_0, ..., x_{2m-1}$

Then the system is cracked, since the linearity of this systems is easy to reverse (thus crack). This is not cryptographically secure.

Proof: Recalling Equation (5.4), by iterating the system for $2n - 1$ number of times we can create a system of $2n$ equations from which Eve can retrieve all the p_i coefficients.

□

5.3.1. Generator polynomial and example

Definition 5.1: A polynomial associated with an LFSR yielding a cycle of maximal period is said to be a **primitive polynomial**.

Remark: The coefficient p_0 must be 1, otherwise it will produce an *antiperiod*: the information in the "FF0" present at the beginning will be lost.

In terms of representative polynomial it means that

$$P(x) = x^m + \sum_{j=1}^{m-1} p_j x^j + 1 \tag{5.5}$$

In order to be associated to a maximal length cycle (i.e. to be a so called **primitive polynomial**) the polynomial P must contain at least three non-trivial terms (the one given by p_0, x^m and another one).

Exercise 5.1 (Attack to LFSR): We want to perform an attack on a LFSR-based stream cipher. Each letter of the alphabet and numbers from 0 to 5 are represented by a 5-bit vector as follows:

$$\begin{aligned} A &\leftrightarrow 0 = 00000_2 \\ &\vdots \\ 5 &\leftrightarrow 31 = 11111_2 \end{aligned} \tag{5.6}$$

We know the following facts about the system:
- The degree of the LFSR is $m = 6$
- Every message starts with the header WPI

We observe on the channel the message: '*j5a0edj2b*'

1. What is the initialization vector?
2. What are the feedback coefficients of the LFSR?
3. Write a program that generates the whole sequence and find the whole plaintext.
4. Where does the thing after 'WPI' live?
5. What type of attack did we perform?

1. We can start by seeing that 'WPI' (22 15 18) gets encoded by 'j5a' (9 31 0)

	W	P	I
'WPI'	1 0 1 1 0	0 1 1 1 1	0 1 0 0 0
'j5a'	0 1 0 0 1	1 1 1 1 1	0 0 0 0 0
Key-stream	**1 1 1 1 1**	**1** 0 0 0 0	0 1 0 0 0

Where the bits of the key stream are obtained by addition.

What we are interested in are first 6 bits of the key-stream, that represent the initialization vector: $(s_0, ..., s_5)$ = (111111)

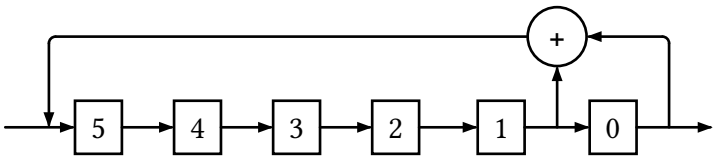
2. Since we know the initialization vector, and also the following bits of the key-stream we can write:

$$\begin{aligned} s_6 &= p_5 \cdot s_5 + p_4 \cdot s_4 + ... + p_0 \cdot s_0 \\ &\Downarrow \\ 0 &= p_5 \cdot 1 \ + p_4 \cdot 1 \ + ... + p_0 \cdot 1 \\ s_7 &= p_5 \cdot s_6 + p_4 \cdot s_5 + ... + p_0 \cdot s_1 \\ &\Downarrow \\ 0 &= p_5 \cdot 0 \ + p_4 \cdot 1 \ + ... + p_0 \cdot 1 \end{aligned} \tag{5.7}$$

By combining the 2nd and 4th expressions we get that $p_5 = 0$. Following a similar process exploiting the rest of the known key-stream, we can retrieve all the other p_i.

The resulting feedback path is (1 1 0 0 0 0) (little-endian) and the polynomial is

$$x^6 + x + 1 \tag{5.8}$$



3. The key-stream is the following:

	IDX
11111 **1**	0
01111 **1**	1
00111 **1**	2
00011 **1**	3
00001 **1**	4
00000 **1**	5
10000 **0**	6
01000 **0**	7
00100 **0**	8
00010 **0**	9
00001 **0**	10

	IDX
10000 **1**	11
11000 **0**	12
01100 **0**	13
00110 **0**	14
00011 **0**	15
10001 **1**	16
01000 **1**	17
00101 **0**	18
10010 **1**	19
11001 **0**	20
11100 **1**	21

	IDX
11110 **0**	22
01111 **0**	23
10111 **1**	24
01011 **1**	25
00101 **1**	26
00010 **1**	27
10001 **0**	28
11000 **1**	29
11100 **0**	30
...	...

In order to decypher we need the keys starting from 15 (since the previous ones are used for WPI), so we get:

KEY-STREAM	01100	01010	01111	01000	11100	10010
'0edj2b'	10110	01110	01100	00001	00000	10011
Plain text	W	O	M	B	A	T

4. It lives in the woods
5. It is a known plain-text attack

5.3.2. Polynomials associated to LFSR

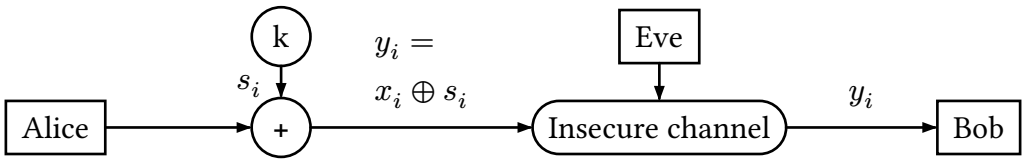
Definition 5.2 (Primitive polynomial): A polynomial associated to with an LFSR yielding a cycle of maximal period is said to be **primitive**.

Definition 5.3 (Irreducible polynomial): A polynomial associated to with an LFSR yielding a cycle of maximal period, but whose length is independent on the initializing state, is said to be **irreducible and not primitive**.

Definition 5.4 (Reducible polynomial): A polynomial associated to with an LFSR yielding a cycle of maximal period and whose length is dependent on the initializing state, is said to be **reducible**.

These features of polynomials are related to the topics of Galois fields (see Section 8. [p. 29])

5.4. One time pad



1. The keystream is generated by a TRNG
2. The keystream is communicated from A to B (or B to A) and it has to be as long as the plaintext.
3. The bits have to be used only once.

Example: 1-bit message

x_i	s_i	y_i
0	0	0
0	1	1
1	0	1
1	1	0

In this case, whatever x_i:

$$y_i = \begin{cases} 0 & \text{with probability } \frac{1}{2} \\ 1 & \text{with probability } \frac{1}{2} \end{cases} \tag{5.9}$$

provided that the TRNG is perfectly balanced.

Theorem 5.1: Consider an arbitrary plaintext (k-bit) message $x \in \underbrace{\mathbb{Z}_2 \times \mathbb{Z}_2 \times ... \times \mathbb{Z}_2}_{k \text{ times}}$

Let $y \in \underbrace{\mathbb{Z}_2 \times \mathbb{Z}_2 \times ... \times \mathbb{Z}_2}_{k \text{ times}}$ be the ciphertext corresponding to x through a OTP encoding. Then the probability of occurrence of y is $P(y) = \frac{1}{2^k}$

Vice-versa given y the probability that a certain x in the same space is the originating plaintext is $P(x) = \frac{1}{2^k}$ (this is the Eve probability point of view)

Proof: Let $x = (x_0, ..., x_{k-1})$ and $y = (y_0, ..., y_{k-1})$ exists one and only one key stream $(s_0, ..., s_{k-1})$ that maps x into y. In fact it is defined by:

$$s_i = x_i \oplus y_i \tag{5.10}$$

Now the key stream given by Equation (5.10) occurs with probability $1/2^k$ provided that a TRNG is used and all S_is are independent of each other and balanced. The proof for the second part is identical. □

Definition 5.5 (Correct encryption scheme): An encryption scheme (e, d) is called **correct** if, for every plaintext x and every key k, we have that $d(e(x,k),k) = x$.

Definition 5.6 (Perfectly secure encryption scheme): It is also called **perfectly secure** if for any distribution P (probability) over the space of the plaintext X, the following operations are identical:

- Generate a random plaintext $x \in X$ with probability $P(x)$
- Consider an arbitrary ciphertext y. Generate a uniformly random key $k \in K$. Generate a random plaintext $x \in X$ with probability $P(x, e(x,k) = y)$

Comment: In the classical One-Time-Pad one has **perfect security**.

5.4.1. A quantum protocol for One-Time-Pad

Idea: Instead of classical bits, consider qubits

$$x = (x_0, ..., x_{k-1}) \mapsto (|x_0\rangle, ..., |x_{k-1}\rangle) = |x_0 ... x_{k-1}\rangle = |x\rangle \tag{5.11}$$

The random generator emits the key stream $S_0 ... S_{k-1}$ that encodes the message in this way:

$$\begin{aligned} &\text{If } s_i = 0 \Rightarrow |x_i\rangle \mapsto |x_i\rangle = \mathbb{I}|x_i\rangle \\ &\text{If } s_i = 1 \Rightarrow |x_i\rangle \mapsto |x_i \oplus 1\rangle = \begin{cases} |1\rangle \text{ if } x_i = 0 \\ |0\rangle \text{ if } x_i = 1 \end{cases} = \sigma_x |x_i\rangle \end{aligned} \tag{5.12}$$

Suppose that the TRNG is quantum, The state of the system made of the TRNG and the i^{th} qubit can be written as

$$\rho = \frac{1}{2} \underbrace{|0\rangle\langle 0|_k \otimes |x_i\rangle\langle x_i|_M}_{\substack{\text{tensor product of two} \\ \text{density matrices}}} + \frac{1}{2} \underbrace{|1\rangle\langle 1|_k \otimes \sigma_x |x_i\rangle\langle x_i|_M \sigma_x}_{\substack{\text{tensor product of two} \\ \text{density matrices}}} \tag{5.13}$$

The first factor is related to the key stream, the second to the message

N.B: This is the density matrix of the encoded message and the key stream

Suppose Eve intercepts the ciphertext, she can have at her disposal nothing but the *reduced density matrix* of the qubit of the message

$$\begin{aligned} \rho_M = \mathrm{Tr}_k(\rho) = |x_i\rangle\langle x_i| + \sigma_x |x_i\rangle\langle x_i| \sigma_x = \\ = \begin{cases} \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{\mathbb{I}}{2} & x_i = 0 \\ \frac{1}{2}|1\rangle\langle 1| + \frac{1}{2}|0\rangle\langle 0| = \frac{\mathbb{I}}{2} & x_i = 1 \end{cases} \end{aligned} \tag{5.14}$$

Whatever M is, Eve gets the same ρ. Supposing that the s_is are equally distributed and independent, for a k-qubit message M one has:

$$\rho = \bigotimes_{i=1}^{k} \left[\frac{1}{2}|0\rangle\langle 0|_k \otimes |x_i\rangle\langle x_i|_M + \frac{1}{2}|1\rangle\langle 1|_k \otimes \sigma_x |x_i\rangle\langle x_i|_M \sigma_x \right]$$
$$\rho_M = \left(\frac{\mathbb{I}}{2}\right)^{\otimes k} = \frac{\mathbb{I}}{2^k} \tag{5.15}$$

A genuinely quantum One-Time-Pad protocol

The main difference between classical and quantum communication lies in the fact that quantum theory is non-commutative

Idea: use both σ_x and σ_z because the first flips the computational basis, while the other flips the Hadamard basis. Consider a key stream made of two bits, k_1, k_2 randomly chosen in $\mathbb{Z}^2$. Now recall that the transformation of the state was

$$\begin{aligned} \rho &\mapsto \frac{1}{2}\mathbb{I}\rho\mathbb{I} + \frac{1}{2}\sigma_x\rho\sigma_x \quad (\text{with } \rho = |x_i\rangle\langle x_i|) \\ &= \frac{1}{2}\sigma_x^0\rho\sigma_x^0 + \frac{1}{2}\sigma_x^1\rho\sigma_x^1 = \frac{1}{2}\sum_{k=0}^{1}\sigma_x^k\rho\sigma_x^k \end{aligned} \tag{5.16}$$

The generalization to the case of a 2-bit key stream can be written as follows

$$\rho \mapsto \frac{1}{4}\sum_{k_1,k_2\in\{0,1\}} \sigma_x^{k_1}\sigma_z^{k_2}\rho\sigma_z^{k_2}\sigma_x^{k_1} \tag{5.17}$$

Example: Consider the case $\rho = \sigma_x$

$$\begin{aligned} &\frac{1}{4}\left(\sigma_x + \overbrace{\sigma_z\sigma_x}^{-\sigma_x\sigma_z}\sigma_z + \overbrace{\sigma_x\sigma_x}^{\mathbb{I}}\sigma_x + \sigma_x\overbrace{\sigma_z\sigma_x}^{-\sigma_x\sigma_z}\sigma_z\sigma_x\right) = \\ &= \frac{1}{4}\left(\sigma_x - \sigma_x\overbrace{\sigma_z\sigma_z}^{\mathbb{I}} + \sigma_x - \overbrace{\sigma_x\sigma_x}^{\mathbb{I}}\overbrace{\sigma_z\sigma_z}^{\mathbb{I}}\sigma_x\right) \\ &= \frac{1}{4}(\sigma_x - \sigma_x + \sigma_x - \sigma_x) = 0 \end{aligned} \tag{5.18}$$

The same reasoning can be applied if:

- $\rho = \sigma_y \quad \rightarrow \quad = 0$
- $\rho = \sigma_z \quad \rightarrow \quad = 0$
- $\rho = \mathbb{I} \quad \rightarrow \quad = \mathbb{I}$

Now every density matrix can be written as: $\rho = \frac{\mathbb{I}}{2} + \frac{1}{2}\left(v_x\sigma_x + v_y\sigma_y + v_z\sigma_z\right)$ This time all bases are effectively encoded but Eve can gain information. For a k-qubit one obtains the same result.

One-Time-Pad protocol:

1. The key (k_1, k_2) is chosen uniformly in $\{0,1\}^2$
2. To encrypt, Alice applies $\sigma_x^{k_1}\sigma_z^{k_2}$
3. To decrypt, Bob applies the inverse $\sigma_z^{k_2}\sigma_x^{k_1}$

5.5. Trivium

Is a combination of three LFSRs as shown in Figure 1. For every register the input is the XOR of two bits:

- The first bit is the output of another register
- The second is a feedback from the same register

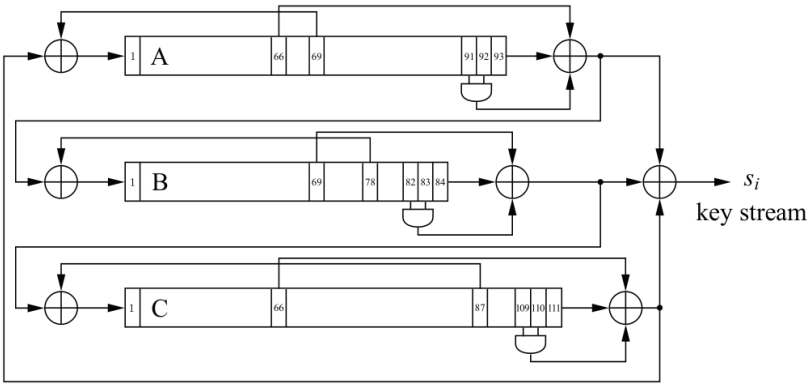


Figure 1: Trivium internal structure

	register length	feedback bit	feedforward bit	AND inputs
A	93	69	66	91, 92
B	84	78	69	82, 83
C	111	87	66	109, 110

Figure 2: Trivium specification

The output of each register is the sum of three bits (specification in Figure 2):

1. The rightmost component of the register
2. A particular bit of the same register that feeds foreward
3. the AND operation carried out on particular bits of the same register

N.B.: The most important feature is the AND, which introduces a non-linearity (quadratic term)

5.5.1. Encryption with trivium

1. Initialization value IV:
   - 80-bit IV into the 80 leftmost bits of register A
   - 80-bit IV into the 80 leftmost bits of register B
   - 3-bit IV into the 3 rightmost bits of register C
2. Warm-up phase: Trivium is clocked 1152 times
3. Encryption phase: from S_{1152} on, the stream ciphers are considered and build up the key stream

6. Block cyphers

Block cyphers are based on two principles by Shannon:

1. **Confusion**: make the connection between the key and the cyphertext as hidden as possible.
2. **Diffusion**: a single bit should influence the ciphertext of other bits. Ideally, changing a bit in the plaintext should result in changes that go beyond the position of that single bit.

In block ciphers, the plaintext is not ciphered bit by bit but block by block so that each bit affects the whole block.

$$\begin{aligned} x_0, ..., x_{k-1} &\mapsto y_0, ..., y_{k-1} \\ x_k, ..., x_{2k-1} &\mapsto y_k, ..., y_{2k-1} \end{aligned} \tag{6.1}$$

6.1. Hill cipher

The Hill cipher is the first historical block cipher, in which each block is ciphered with another block, e.g. $AA \mapsto WY, AB \to OA, ..., ZZ \mapsto RB$.

If each block is composed of 2 letters of the alphabet, the substitution table is made of $26 \cdot 26 = 676$ elements, for block of 3 letters we would have 17476 elements, too many to memorize. To make the encryption easier, we can use a matrix K with m rows and m columns.

$$K \begin{pmatrix} x_0 \\ \vdots \\ x_{m-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_{m-1} \end{pmatrix} \tag{6.2}$$

Diffusion is given by non diagonal elements of the matrix, which tell how a bit affects the other bits in the block.

Exercise 6.1 (Encoding the word *'snow'*):

$$\begin{aligned} K &= \begin{pmatrix} 4 & 3 \\ 1 & 2 \end{pmatrix} \\ \text{"}sn\text{"} &\mapsto \begin{pmatrix} 18 \\ 13 \end{pmatrix} = \vec{x_1} \\ \text{"}ow\text{"} &\mapsto \begin{pmatrix} 14 \\ 22 \end{pmatrix} = \vec{x_2} \end{aligned} \tag{6.3}$$

$$\begin{aligned} \vec{y_1} = K\vec{x_1} = \begin{pmatrix} 4 & 3 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 18 \\ 13 \end{pmatrix} = \begin{pmatrix} 4 \cdot 18 + 13 \cdot 3 \bmod 26 \\ 1 \cdot 18 + 2 \cdot 13 \bmod 26 \end{pmatrix} = \begin{pmatrix} 7 \\ 18 \end{pmatrix} = \text{"}HS\text{"} \\ \vec{y_2} = K\vec{x_2} = ... = \text{"}SG\text{"} \end{aligned} \tag{6.4}$$

Note that diffusion is limited inside the block.

In order to decrypt "HSSG", Bob needs to invert the matrix K, i.e. find B such that $BK = \mathbb{I}$.

Since we are working in modular algebra, it is **not true** that

$$\text{matrix is invertible} \Leftrightarrow \det(\cdot) \neq 0 \tag{6.5}$$

Example (Matrix with no inverse): We now consider the matrix

$$K' = \begin{pmatrix} 3 & 1 \\ 1 & 1 \end{pmatrix} \tag{6.6}$$

wich has determinant $\det(K') = 2 \neq 0$.

Suppose that $\exists B$ s.t. $BK' = \mathbb{I}$:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \\ & 1 \end{pmatrix}$$

$$\begin{pmatrix} 3a+b & a+b \\ 3c+d & c+d \end{pmatrix} = \begin{pmatrix} 1 & \\ & 1 \end{pmatrix}$$

$$\begin{cases} 3a+b=1 & (\heartsuit) \\ a+b=0 & (\diamondsuit) \\ 3c+d=0 \\ c+d=1 \end{cases} \tag{6.7}$$

$$\begin{cases} \heartsuit \\ \diamondsuit \end{cases} \Longrightarrow 2a = 1 \bmod 26 \Longrightarrow \text{no solution since} \gcd(2,26) = 2 \neq 1$$

This implies that B does not exists and thus that K' has **no inverse in modular algebra**.

Another problem is that the matrix must be **injective**, otherwise two different block may be mapped to the same block, making decryption impossible. This condition, in linear algebra, as well as in modular algebra, corresponds to the matrix being invertible.

Theorem 6.1 (Hill):

Consider the matrix $K = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with elements in $\mathbb{Z}_{26}$, then the following statements are equivalent:

1. $ad - bc (\equiv \det(k))$ is not divisible by neither 2 nor 13 (6.8)
2. $\exists B$, a 2×2 matrix with elements in $\mathbb{Z}_{26}$ s.t. $BK = \mathbb{I}$ (6.9)
3. K satisfies: $\vec{x_1} \neq \vec{x_2} \Rightarrow K\vec{x_1} \neq K\vec{x_2}$ (Injective) (6.10)

Proof: We have to prove that $1) \Longrightarrow 2) \Longrightarrow 3) \Longrightarrow 1)$, we do this in three steps:

- $1) \Longrightarrow 2)$

$ab - cd$ is not divisible by 2 or 13 $\Longrightarrow \exists h \in \mathbb{Z}_{26}$ s.t. $h(ad - bc) = 1 \bmod 26$

Let $B \triangleq \begin{pmatrix} hd & -hb \\ -hc & ha \end{pmatrix}$ then

$$BK = h\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = h\begin{pmatrix} ad-bc & \cancel{db-bd} \\ \cancel{-ca+ca} & ad-bc \end{pmatrix} = h(ad-bc)\mathbb{I} = \mathbb{I} \quad (6.11)$$

- $2) \implies 3)$

We suppose $\exists B$ s.t. $BK = \mathbb{I}$, if $k\vec{x_1} = k\vec{x_2} \implies \underbrace{BK}_{\mathbb{I}}\vec{x_1} = \underbrace{BK}_{\mathbb{I}}\vec{x_2} \implies \vec{x_1} = \vec{x_2}$.

Conversely, if $\vec{x_1} \neq \vec{x_2} \implies B\vec{x_1} \neq B\vec{x_2}$

- $3) \implies 1)$

Suppose that $\vec{x_1} \neq \vec{x_2} \Rightarrow K\vec{x_1} \neq K\vec{x_2}$, we want to prove that $ad - bc$ is not divisible neither by 13 nor by 2. Suppose that $2 \mid (ad - bc)$, then $13(ad - bc) = 0 \bmod 26$. Then consider

$$K\begin{pmatrix} 13d \\ -13c \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} 13d \\ -13c \end{pmatrix} = 13\begin{pmatrix} ad-bc \\ cd-cd \end{pmatrix} = 13\underbrace{\begin{pmatrix} ad-bc \\ 0 \end{pmatrix}}_{\text{♥}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (6.12)$$

Analogously

$$K\underbrace{\begin{pmatrix} -13b \\ 13a \end{pmatrix}}_{\text{♦}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (6.13)$$

Obviously, the two vectors ♥ and ♦ cannot be both $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, otherwise we would have $K = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

But $K\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ implies that there are two vectors ($\begin{pmatrix} 0 & 0 \end{pmatrix}^\mathsf{T}$ and at least one among ♥ and ♦) contradicting "3)".

The same contradiction can arise from the hypothesis $13|(ad - bc)$ □

Theorem 6.2 (Generalization of Hill): For a Hill cypher with an alphabet of N letters, block size of m and a key matrix K, with elements in $\mathbb{Z}_N$, the following statements are equivalent:

1. $\det(k)$ and N are coprime (6.14)
2. $\exists B$ an $m \times m$ matrix s.t. $BK = \mathbb{I}$ (6.15)
3. $\vec{x_1} \neq \vec{x_2} \Rightarrow k\vec{x_1} \neq k\vec{x_2}$ (6.16)

Exercise 6.2:

1) With the K used to encode *"snow"* (Exercise 6.1 [p. 19]) decrypt "HSSG" as if you were Bob.

2) Using $K_2 = \begin{pmatrix} 25 & 0 \\ 2 & 1 \end{pmatrix}$
   a) Encode *"four"*.
   b) Let $\alpha_1\alpha_2\alpha_3\alpha_4$ be the answer of a). Encode it with K_2 again.
   c) Explain
3) Bob receives "SMKH" from Alice with $K_3 = \begin{pmatrix} 2 & 1 \\ 3 & 6 \end{pmatrix}$, what is the plaintext?

1)

$$K = \begin{pmatrix} 4 & 3 \\ 1 & 2 \end{pmatrix}, \det(K) = 5 \to K^{-1} = 21 \cdot \begin{pmatrix} 2 & -3 \\ -1 & 4 \end{pmatrix} = \begin{pmatrix} -10 & 15 \\ 5 & 6 \end{pmatrix}$$

$$\text{"}HS\text{"} \mapsto \begin{pmatrix} 7 \\ 18 \end{pmatrix} = \vec{x_1}$$

$$\text{"}SG\text{"} \mapsto \begin{pmatrix} 18 \\ 6 \end{pmatrix} = \vec{x_2} \tag{6.17}$$

$$\vec{y_1} = K^{-1}\vec{x_1} = \begin{pmatrix} -10 & 15 \\ 5 & 6 \end{pmatrix}\begin{pmatrix} 7 \\ 18 \end{pmatrix} = \begin{pmatrix} -70 + 15 \cdot 18 \bmod 26 \\ 35 - 48 \bmod 26 \end{pmatrix} = \begin{pmatrix} 18 \\ 13 \end{pmatrix} = \text{"}sn\text{"}$$

$$\vec{y_2} = K\vec{x_2} = ... = \text{"}ow\text{"}$$

$$\text{"}HSSG\text{"} \mapsto \text{"}snow\text{"}$$

2)

a)

$$K = \begin{pmatrix} 25 & 0 \\ 2 & 1 \end{pmatrix}$$

$$\text{"}fo\text{"} \mapsto \begin{pmatrix} 5 \\ 14 \end{pmatrix} = \vec{x_1}$$

$$\text{"}ur\text{"} \mapsto \begin{pmatrix} 20 \\ 17 \end{pmatrix} = \vec{x_2} \tag{6.18}$$

$$\vec{y_1} = K\vec{x_1} = \begin{pmatrix} 25 & 0 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} 5 \\ 14 \end{pmatrix} = \begin{pmatrix} (25 \cdot 5) \bmod 26 \\ 10 + 14 \bmod 26 \end{pmatrix} = \begin{pmatrix} 21 \\ 24 \end{pmatrix} = \text{"}WY\text{"}$$

$$\vec{y_2} = K\vec{x_2} = ... = \text{"}GF\text{"}$$

$$\text{"}four\text{"} \mapsto \text{"}WYGF\text{"}$$

b) We can notice that K_2 is the inverse of itself since

$$\begin{aligned} K_2 \cdot K_2 &= \begin{pmatrix} 25 & 0 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 25 & 0 \\ 2 & 1 \end{pmatrix} \\ 25 \equiv -1 \bmod 26 \longrightarrow &= \begin{pmatrix} (-1 \cdot -1 + 0) & 0 \\ 0 & (2 \cdot -1 + 1) \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{aligned} \tag{6.19}$$

So encoding twice with K_2 is the same as encoding and decoding which is to say doing nothing.

3) We start by computing the inverse of K_3:

$$K_3^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$K_3 \cdot K_3^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\Downarrow$$

$$\begin{cases} 2a + c = 1 \\ 2b + d = 0 \\ 3a + 6c = 0 \\ 3b + 6d = 1 \end{cases} \Longrightarrow \begin{cases} a = 18 \\ b = 23 \\ c = 17 \\ d = 6 \end{cases} \tag{6.20}$$

$$K_3^{-1} = \begin{pmatrix} 18 & 23 \\ 17 & 6 \end{pmatrix}$$

The rest of the exercise is the same as the one done before, simply encode the words into vectors and perform the matrix-vector multiplication. The result is *"code"*.

Exercise 6.3 (Attack to Hill cypher 1): You have intercepted the message "WGTK" and know it has been encrypted using a Hill cipher. You know also that "cd" is encrypted as "RR" and "jk" is encrypted as "OV", what is the plaintext?
As a first thing we need to find the matrix for the encryption by constructing the system of equations given by $cd \to RR$ and $jk \to OV$:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 17 \\ 17 \end{pmatrix} \text{ and } \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 9 \\ 10 \end{pmatrix} = \begin{pmatrix} 14 \\ 21 \end{pmatrix} \tag{6.21}$$

This leads to:

$$\begin{cases} 2a + 3b = 17 \\ 2c + 3d = 17 \\ 9a + 10b = 14 \\ 9c + 10b = 21 \end{cases} \tag{6.22}$$

Since we are working in modular algebra in order to find the elements of the matrix is easier to multiply their coefficients that have an inverse in mod 26 by this inverse:

$$\begin{gathered} 9 \times \begin{cases} 2a + 3b = 17 \\ 2c + 3d = 17 \end{cases} \to \begin{cases} 18a + b = 23 = -3 \\ 18c + d = 23 = -3 \end{cases} \Rightarrow \begin{cases} b = -3 - 18a \\ d = -3 - 18c \end{cases} \\ \begin{cases} 9a + 10(-3 - 18a) = 14 \\ 9c + 10(-3 - 18c) = 21 \end{cases} \Rightarrow \begin{cases} 11a = 18 \\ 11c = 25 \end{cases} \to 19 \times \begin{cases} 11a = 18 \\ 11c = 25 \end{cases} \Rightarrow \begin{cases} a = 56 = 4 \\ c = -19 = 7 \end{cases} \end{gathered} \tag{6.23}$$

And substituting a and c in the previous relations we find $b = 3$ and $d = 1$ so the matrix k is:

$$\begin{pmatrix} 4 & 3 \\ 7 & 1 \end{pmatrix} \tag{6.24}$$

To decrypt the message we need now to find the inverse. $\det(K) = 4 - 21 = -17 = 9$ and its inverse is 3

$$K^{-1} = 3 \cdot \begin{pmatrix} 1 & -3 \\ -7 & 4 \end{pmatrix} = \begin{pmatrix} 3 & -9 \\ 5 & 12 \end{pmatrix} \tag{6.25}$$

Now we decrypt by applying K^{-1} to the vectors corresponding to “WGTK”, that are $(22, 6)^T$ and $(19, 10)^T$:

$$\begin{aligned} \begin{pmatrix} 3 & -9 \\ 5 & 12 \end{pmatrix} \begin{pmatrix} 22 \\ 6 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \end{pmatrix} = \begin{pmatrix} M \\ A \end{pmatrix} \\ \begin{pmatrix} 3 & -9 \\ 5 & 12 \end{pmatrix} \begin{pmatrix} 19 \\ 10 \end{pmatrix} = \begin{pmatrix} 19 \\ 7 \end{pmatrix} = \begin{pmatrix} T \\ H \end{pmatrix} \end{aligned} \tag{6.26}$$

7. DES Algorithm

DES is a symmetric (i.e. the same key is used both for encryption and decryption) cipher that encrypts blocks of length of 64 bits with a key of size 56 bits.

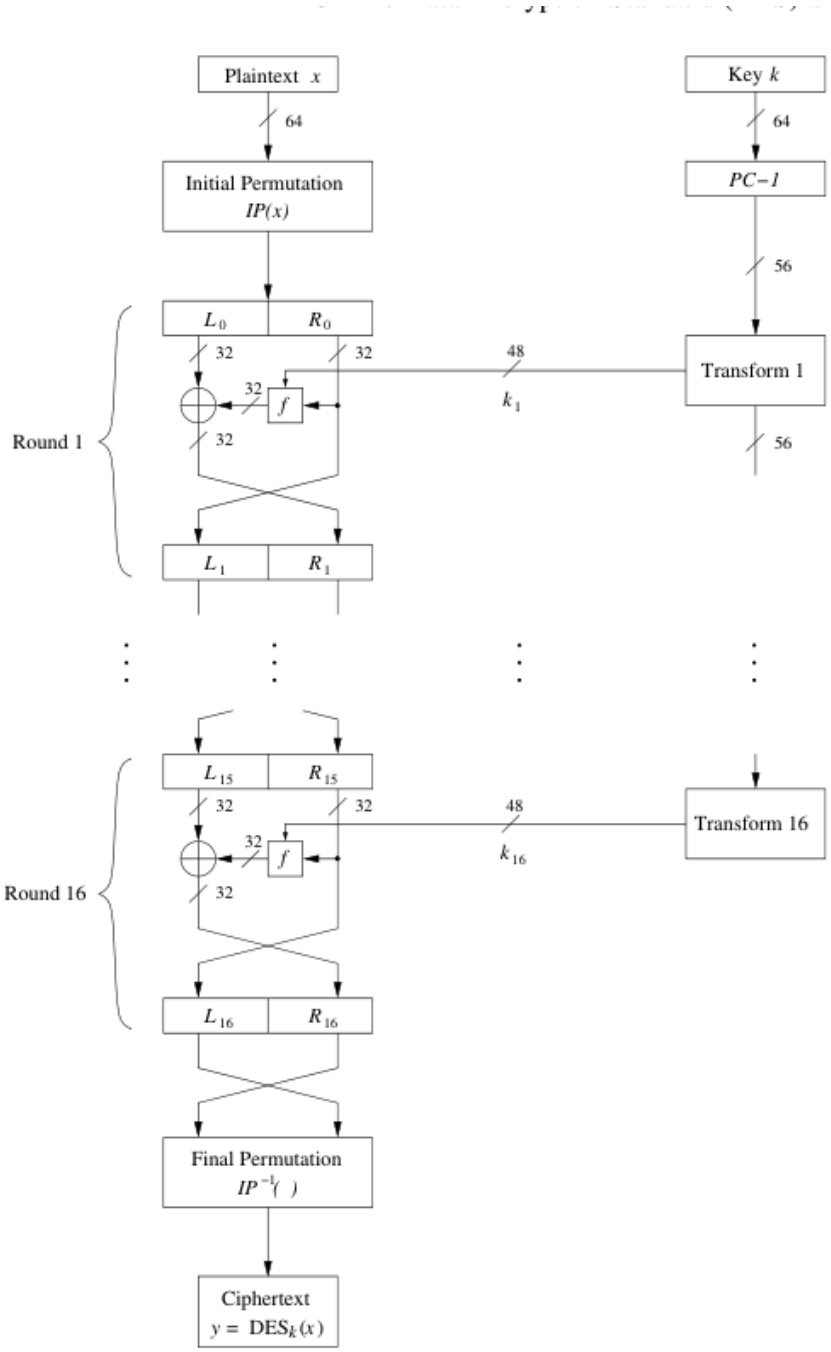


Figure 3: DES Feistel structure

The IP and the IP^{-1} are two bitwise permutations, the IP is the first operation performed on the plaintext, while the IP^{-1} is the last operation being performed to obtain the ciphertext.

The way to read the tables is simply to read it cell by cell and take the bit corresponding to the current element in the cell and place in the current position. Reading IP, for example, we have to place bit 58 in the 1st position, bit 50 in the 2nd and so on.

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Fig. 3.8 Initial permutation IP

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Fig. 3.9 Final permutation IP^{-1}

Figure 4: DES initial and final permutations

7.0.1. The f function

This function plays a crucial role for the security of DES. In the i-th round it takes the right half of the output of the previous round and the current key k_i as inputs; the result of the f function is used as a XOR-mask for encrypting the $L_{\{i-1\}}$ part.

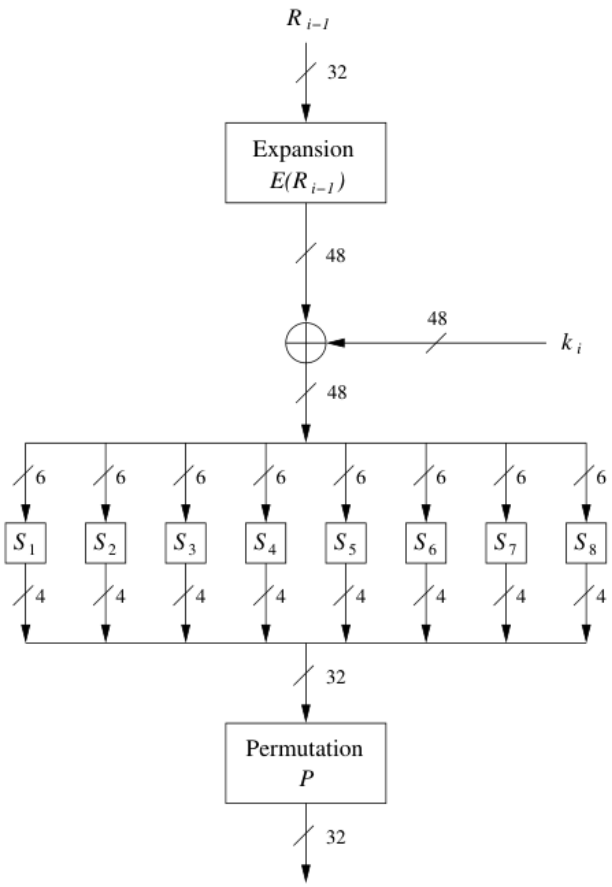


Figure 6: f-function block scheme

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Figure 5: Expansion matrix

P							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Figure 7: Permutation matrix

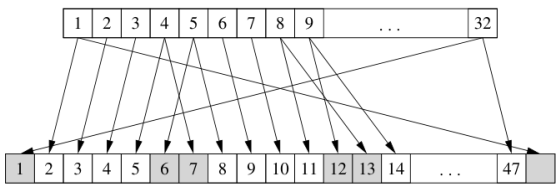


Figure 8: Bit swaps for the Expansion function

The expansion table is to be read similarly to the IP table, noting that some of the numbers appear more than once in the table. As a first thing the 32-bit input is expanded and the bits are permutated as shown in Figure 6 [p. 26]. Then the 48-bit result of the expansion is XORed with the key k_i and the eight 6-bit blocks of the result are fed into eight *Substitution boxes* (or **S-boxes**). Each box maps the 6-bit input into a 4-bit output and every *S-box* is unique. The mapping is done by reading in the 6-bit input the row and the column correspondent to the position in the table of the referred *S-box*, and finding in the position the decimal number corresponding to the 4-bit output. In the example in Figure 9, we use the first and last bit to get the index of the row (in this case $(11)_2 = 3$, which means we the the row with index 3; the same is done to get the column number using the remaining bits.

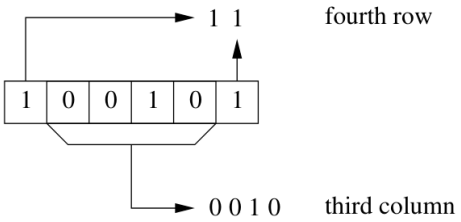


Figure 9: Example of the decoding of the input $(100101)_2$ for the *S-box*

The matrices corresponding to the *S-boxes* are omitted in this notebook for the sake of brevity. The *S-boxes* are designed following multiple criteria in order to achieve a high cryptographic strength. The importance of the *S-boxes* is the *non-linearity* they introduce to the cipher:

$$S(a) \oplus S(b) \neq S(a \oplus b) \tag{7.1}$$

The permutation P in the f-function introduces diffusion, affecting various *S-boxes* in the following round. The diffusion given by the permutation, the *S-boxes* and the expansion is called **avalanche effect**.

7.0.2. Key schedule

$PC-1$							
57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4

Figure 10: $PC-1$ table

$PC-2$							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Figure 11: $PC-2$ table

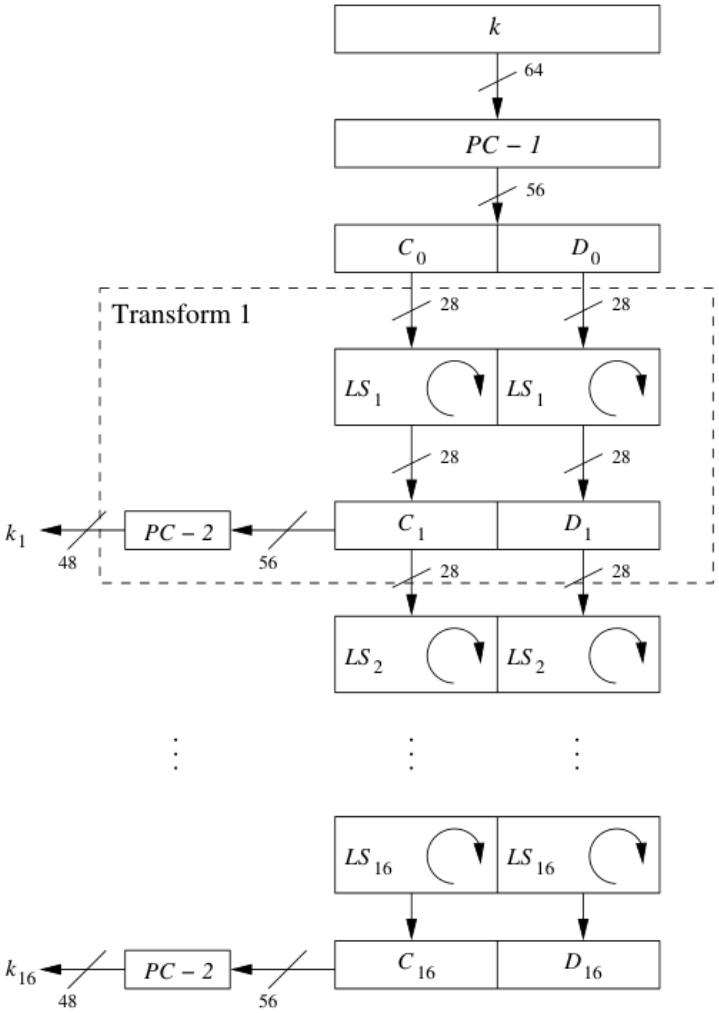


Figure 12: DES key schedule block diagram

The key schedule in DES derives 16 round keys k_i, each consisting of 48 bits, from the original 56-bit key. the input key is 64-bit long, where every eighth bit is used as an odd parity bit over the preceding seven bits. Thus those parity bits are ignored, they are stripped in the initial $PC-1$ permutation. The resulting 56-bit key is halved in C_0 and D_0 and the key schedule starts as shown in Figure 12 and the two parts are cyclically shifted left (LS_i) following the rule:

- In rounds $i = 1, 2, 9, 16$ the two halves are rotated **left by one bit**.
- In the other rounds ($i \neq 1, 2, 9, 16$) the two halves are rotated **left by two bits**.

To derive each sub-key the two halves are permuted bitwise again but with the permutation $PC-2$.

7.0.3. Decryption

One of the advantages of DES is that decryption is essentially the same function as encryption: only the key schedule is reversed i.e. in decryption round 1 the subkey k_{16} is needed. Thus in decryption the key schedule algorithm has to generate the round keys as the sequence $k_{16}, k_{15}, ..., k_1$.

With the initial key k it is possible to obtain directly k_{16}:

$$\begin{aligned} k_{16} &= PC-2(C_{16}, D_{16}) \text{ but since } \ C_{16} = C_0 \text{ and } C_{16} = C_0 \\ &= PC-2(C_0, D_0) \\ &= PC-2(PC-1(k)) \end{aligned} \tag{7.2}$$

8. Galois Fields (Theory of finite fields)

Take a look at Rings, Section 3.5. [p. 9]

8.0.1. Properties with respect to sum

$$\begin{cases} (a+b)+c = a+(b+c) = a+b+c \\ a+b = b+a \\ \exists 0 \in \mathbb{Z}_p \text{ s.t. } a+0 = 0+a = a \quad \forall a \\ \forall a \in \mathbb{Z}_p \exists b \in \mathbb{Z}_p \text{ s.t. } b+a = 0 \end{cases} \tag{8.1}$$

$$\implies (\mathbb{Z}, +) \text{ is a commutative group}$$

8.0.2. Properties with respect to product

$$\begin{cases} (a \cdot b)c = a \cdot (b \cdot c) \\ a \cdot b = b \cdot a \\ \exists \mathbb{1} \in \mathbb{Z}_p \text{ s.t. } a\mathbb{1} = \mathbb{1}a = a \quad \forall a \\ a \cdot (b+c) = a \cdot b + a \cdot c \end{cases} \tag{8.2}$$

$$\implies (\mathbb{Z}, \cdot) \text{ is NOT a group (due to lack of the inverse)}$$

$(\mathbb{Z}, +, \cdot)$ is a **commutative ring with unity**.

Notice that one of the differences between rings and groups is that rings have **two** operations while groups have only **one** operation.

8.1. Examples

Example: Considering $\mathbb{Z}_5$:

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

We generate the whole group ciclically: $\mathbb{Z}_5 = \{a + x, x \in \mathbb{Z}_5\}$

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$\mathbb{Z}_5 = \{a \cdot x, x \in \mathbb{Z}_5\}$ except for the first row

Example: Now we do the same for $\mathbb{Z}_6$:

$+$	0	1	2	3	4	5
0						
1						
2			... As before ...			
3						
4						
5						

$\cdot$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Now we don't get all the elements

Observations:

1. Some lines do not contain the whole set
2. This is not a **domain of integrity** (we get 0 when neither of the elements are 0)
3. When an element is missing in a line, also "1" is missing

The three phenomena above do not occur if m is prime. In this case we also get property 4. for multiplication:

$$\forall a \in \mathbb{Z}_p \setminus \{0\} \quad \exists a^{-1} \text{ s.t. } a^{-1}a = \mathbb{1} \tag{8.3}$$

Definition 8.1 (Field): If all the axioms in Equation (8.1) [p. 29], Equation (8.2) [p. 29] and Equation (8.3) are satisfied, we say that the set is a **field**.

Theorem 8.1 (Fields): $\mathbb{Z}_m$ is a field $\iff$ m is prime

How many elements can a finite field have?

Theorem 8.2: Given $m \in \mathbb{N}$, a field with m elements exists iff $m = p^n$ with p prime and $n \in \mathbb{N}$

Example: $128 = 2^7 \implies \exists$ a finite field with 128 elements

Theorem 8.3: Given $m = p^k$ (p prime) there is ("essentially") <u>one</u> finite field with m elements.
It is denoted by $\mathrm{GF}(m)$

Finite fields are split into 2 categories:

- **Prime fields:** with $m = p$, they are the $\mathbb{Z}_p$
- **Extension fields:** with $m = p^n, n > 2$

8.2. Structure of $GF(2^m)$

Intuition: consider m copies of $\mathbb{Z}_2$. We can imagine to construct a polynomial with a_i as the coefficients:

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_1 x + a_0 \quad a_i \in \mathbb{Z}^2 \tag{8.4}$$

We want to define the **sum** and **product** operations:

Definition 8.2 (Sum in extension fields $GF(2^m)$):

$$\begin{gathered} A(x) = \sum_{j=0}^{m-1} a_j x^j \quad B(x) = \sum_{j=0}^{m-1} b_j x^j \\ A(x) + B(x) = \sum_{j=0}^{m-1} \big(a_j \oplus b_j\big) x^j \end{gathered} \tag{8.5}$$

For the product it is not as easy since the classical polynomial product is not closed (because when doing the classical product, we sum the exponents and this can cause them to become $> m$). We can "close" the multiplication by taking the modulo with respect to an irreducible polynomial (Definition 5.3 [p. 14]).

Definition 8.3 (Product in extension fields $GF(2^m)$):
Let $P(x) = \sum_{j=0}^{m-1} p_j x^j, \quad p_i \in \mathbb{Z}_2 = GF(2)$ be an irreducible polynomial, then:

$$C(x) = A(x) \cdot B(x) \bmod P(x) \tag{8.6}$$

Exercise 8.1: Compute $A(x)B(x) \bmod P(x)$ given

$$\begin{gathered} A, B \in \mathrm{GF}(2^4) \\ A(x) = x^3 + x^2 + 1 \\ B = x^2 + x \\ P(x) = x^4 + x + 1 \notin \mathrm{GF}(2^4) \end{gathered} \tag{8.7}$$

$$A(x)B(x) = x^5 + \cancel{x^4 + x^4} + x^3 + x^2 + x = x^5 + x^3 + x^2 + x \tag{8.8}$$

We now have to perform polynomial division:

x^5	$+x^3$	$+x^2$	$+x$	x^4	$+x$	$+1$
x^5	$+0$	$+x^2$	$+x$	x		
	$\boldsymbol{x^3}$					

This means that

$$\underbrace{x^5+x^3+x^2+x}_{a} = \underbrace{x^4+x+1}_{m}\cdot\underbrace{x}_{k}+\underbrace{x^3}_{r} \Longrightarrow \underbrace{AB}_{a} = \underbrace{x^3}_{r} \bmod \underbrace{P}_{m} \tag{8.9}$$

8.2.1. Inversion in $GF(2^m)$

(This is the operation (with $m = 8$) involved in AES)

Find A^{-1} s.t. $A^{-1}(x)A(x) = 1 \bmod P(x)$ is difficult.

Example: Find the inverse of $A(x) = x^7 + x^6 + x$ with $P = x^8 + x^4 + x^3 + x + 1$ (which is the AES polynomial).

1. Write A as a binary string:
   - $A = \left(\underbrace{1100}_{\text{x = C}}\ \underbrace{0010}_{\text{y = 2}}\right) = (C2)_{\text{hex}}$
2. Go to the table in Figure 15 [p. 35] at row C, column 2 and you will find $2F = (0010\ 1111) = x^5 + x^3 + x^2 + x + 1$

So $(x^7 + x^6 + x)(x^5 + x^3 + x^2 + x + 1) = 1 \bmod P(x)$

Exercise 8.2 (Inverting a polynomial): Compute the inverse of $B(x) = x^2 + x$ in $\mathrm{GF}(2^4)$ using $P(x) = x^4 + x + 1$ as the inverting polynomial.

We will use the algorithm described in Section 10.4.1. [p. 45].

$x^4 + x + 1$	0	1	
$x^2 + x$	1	0	
$1 =$ $(x^4 + x + 1) - (x^2 + x)$	$\underline{x^2 + x + 1} =$ $0 - 1 \cdot x^2 + x + 1$	$1 =$ $1 - 0 \cdot x^2 + x + 1$	$x^2 + x + a =$ $\lfloor x^4 + x + 1/x^2 + a \rfloor$

We can see that $B^{-1}(x) = x^2 + x + 1$.

9. AES (Advanced Encryption Standard)

9.1. Properties

1. Must support key sizes of 128, 192, 256 bits
2. The number of rounds is related to the key size: the longer the key the more the rounds.

9.2. Internal structure of AES

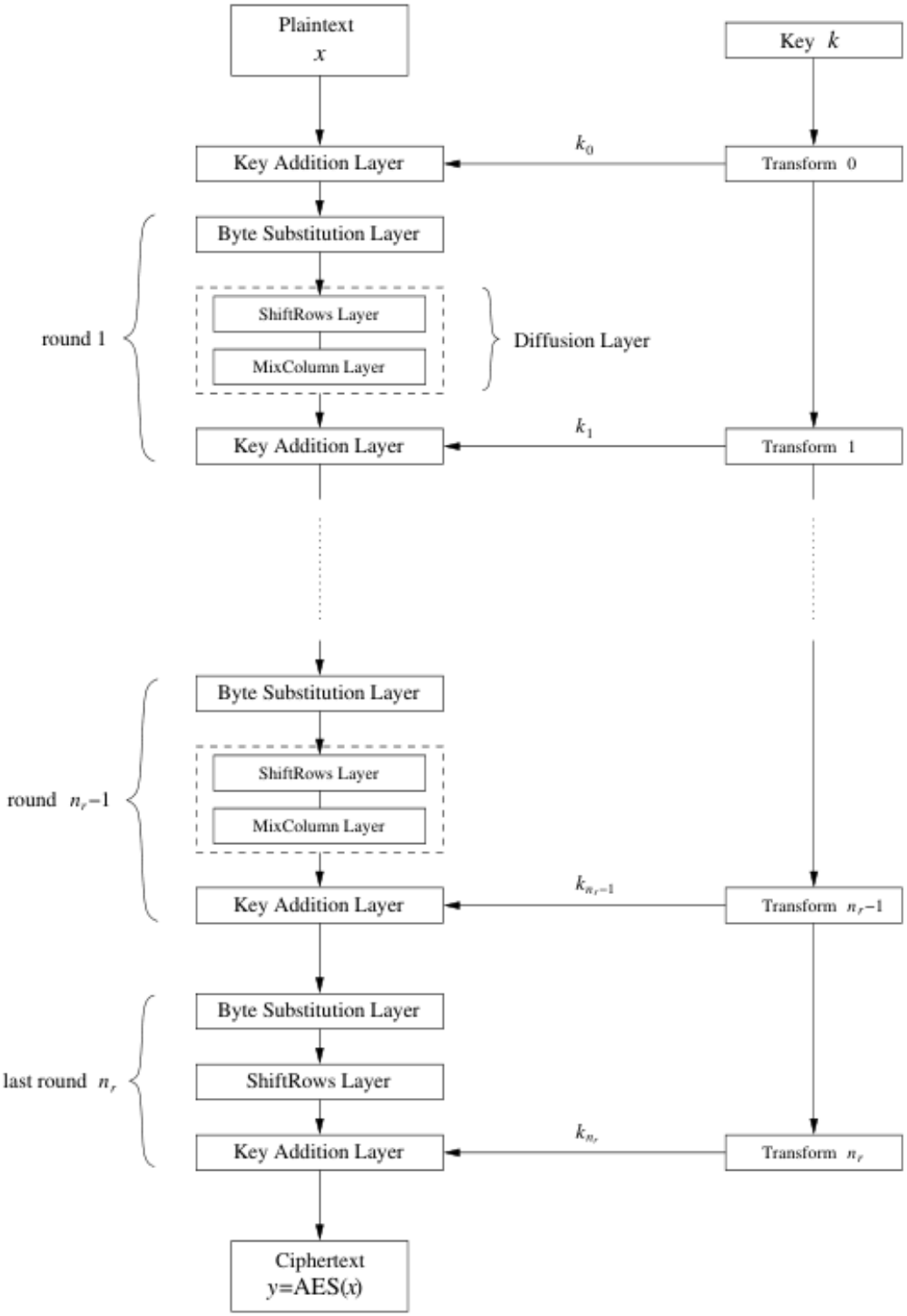


Figure 13: AES block diagram

In the *Byte Substitution Layer* we need the knowledge of *Galois Fields* (Section 8. [p. 29])

AES is a *byte-wise* algorithm, in the case of $x = 128$, the 16 bytes are grouped in groups of 4 bytes.

9.2.1. Key addition layer

The inputs are the current 16-byte state matrix and a 16-bytes sub-key, they are combined through as XOR operation (XOR in the Galois fields is equivalent to the sum).

9.2.2. Key schedule

This process takes the original input key and derives the 16 sub-keys. Note that in AES the addition of a sub-key is used bot at the input and output: this is called *key whitening*.

A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

Table 2: Input state A as a matrix

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

Table 3: state matrix of a 128-bit key

k_0	k_4	k_8	k_{12}	k_{16}	k_{20}
k_1	k_5	k_9	k_{13}	k_{17}	k_{21}
k_2	k_6	k_{10}	k_{14}	k_{18}	k_{22}
k_3	k_7	k_{11}	k_{15}	k_{19}	k_{23}

Table 4: state matrix of a 192-bit key

9.2.3. Byte substitution layer

Every byte undergoes the transformation *S*, which is done by 16 *S-boxes*. In contrast to DES, these *S-boxes* are all identical. In the layer each byte state A_i is substituted with another byte B_i. The *S-boxes* are the only non-linear element of AES, the substitution is a bijective mapping, which allows to uniquely reverse the *S-box* for decryption.

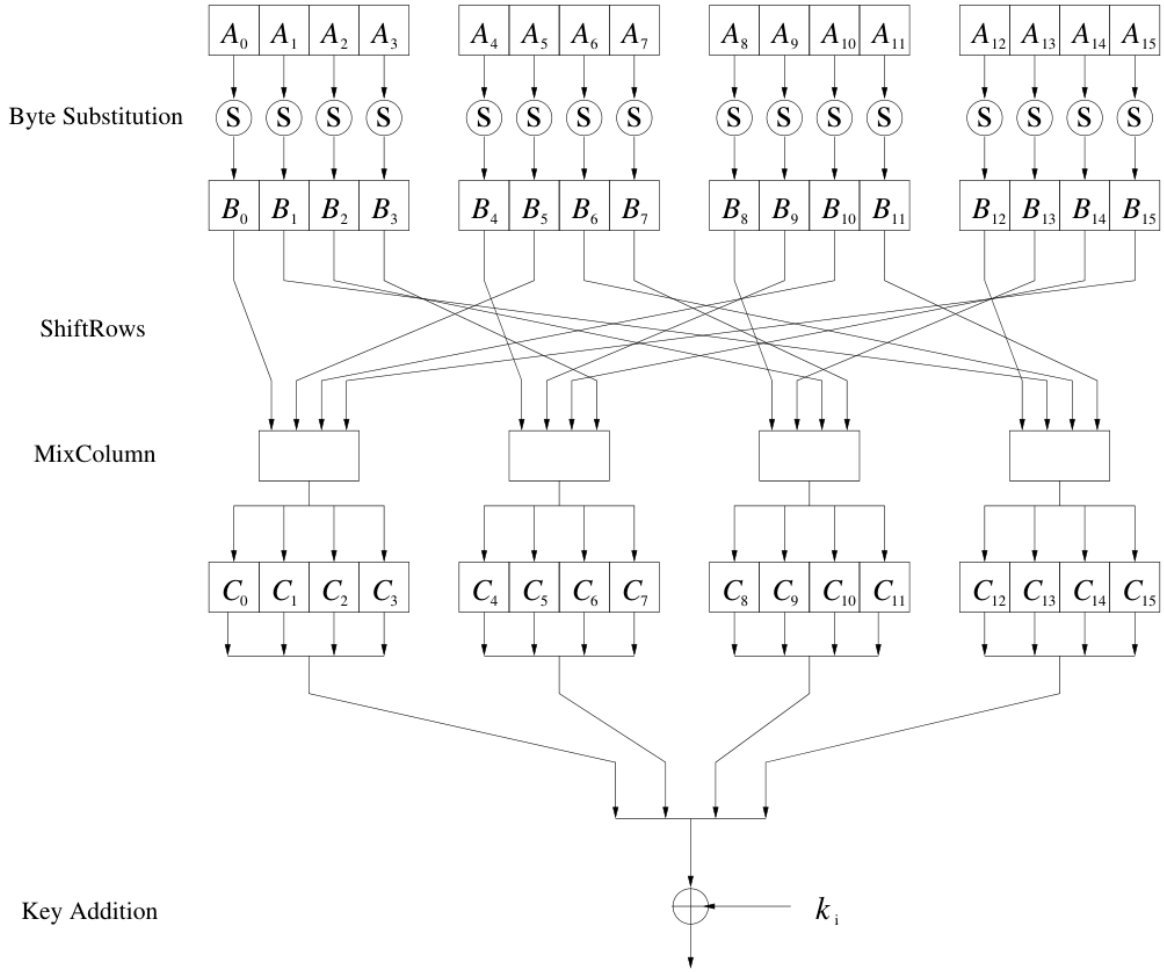


Figure 14: AES round block scheme

X \ Y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

Figure 15: Multiplicative inverse table in $GF(2^8)$ used in AES S-box

The *S-boxes* can be seen as a two-step mathematical transformation, the first one is a *Galois field inversion* (see Section 8.2.1. [p. 32]). For each A_i the inverse is computed, obtaining the byte B'_i, with the fixed irreducible polynomial $P(x) = x^8 + x^4 + x^3 + x + 1$. A lookup table for all the inverse is reported in Figure 15. The inverse of zero is not defined, but in AES $A_i = 0$ is mapped to itself.

In the second part of the substitution each byte undergoes an *affine mapping*, described by:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \bmod 2 \tag{9.1}$$

Example: Assume the input $A_i = (C2)_{\text{hex}} = (1100\ 0010)_2$

From the table in Figure 15 [p. 35] we can see that the inverse is

$$A_i^{-1} = B'_i = (2F)_{\text{hex}} = (0010\ 1111)_2 \tag{9.2}$$

We now apply the affine transformation to B'_i

$$B_i = (0010\ 0101)_2 = (25)_{\text{hex}} \tag{9.3}$$

9.2.4. Diffusion layer

This layer consists of two sublayers: the *ShiftRows* transformation and the *MixColumn* transformation, these perform a linear operation.

ShiftRows

The state matrix is shifted as reported below, following the rules:

- The first row is left untouched
- The second row is shifted right of three positions
- The third row is shifted right of two positions
- The fourth row is shifted right of one position

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

Table 5: Input state

B_0	B_4	B_8	B_{12}	← No shift
B_5	B_9	B_{13}	B_1	← One position shift left
B_{10}	B_{14}	B_2	B_6	← Two positions shift left
B_{15}	B_3	B_7	B_{11}	← Three positions shift left

Table 6: Output state

MixColumns

This linear transformation mixes each column of the state matrix. This is a major diffusion element in AES. The transformation takes as input the state B after the *ShiftRows*.

$$\text{MixColumn}(B) = C \tag{9.4}$$

Each 4-byte column is considered as a vector and multiplied by a fixed 4×4 matrix. The multiplication and addition of coefficients is done in $GF(2^8)$

Example: Calculation of the first four output bytes

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix} \tag{9.5}$$

The other columns of bytes are computed by multiplying the correspondent column of the input state by the same matrix

Example: Calculation of the coefficient C_0 for the first column.

Assuming that the input state to the *MixColumn* layer is $B = (25, 25, ..., 25)$

since the input column has only the value $(25)_{\text{hex}}$ only two multiplication have to be computed: $02 \cdot 25$ and $03 \cdot 25$. They can be computed in polynomial notation. Notice that $(25)_{\text{hex}} = 2^5 + 2^2 + 2^0$.

$$\begin{aligned} 02 \cdot 25 &= x \cdot (x^5 + x^2 + 1) \\ &= x^6 + x^3 + x, \\ 03 \cdot 25 &= (x+1)x^5 + \dot{x}^2 + 1 \\ &= (x^6 + x^3 + x) + (x^5 + x^2 + 1) \\ &= x^6 + x^5 + x^3 + x^2 + x + 1 \end{aligned} \tag{9.6}$$

Since both have a degree smaller than 8, no modular reduction with $P(x)$ is necessary.

The output bytes of C result from the following addition in $GF(2^8)$:

$$\begin{array}{rcccccccc} 01 \cdot 25 = & & & x^5 + & & x^2 + & & 1 \\ 01 \cdot 25 = & & & x^5 + & & x^2 + & & 1 \\ 02 \cdot 25 = & x^6 + & & x^3 + & & x + & & 0 \\ 03 \cdot 25 = & x^6 + & x^5 + & x^3 + & x^2 + & x + & & 1 \\ \hline C_0 = & & & x^5 + & & x^2 + & & 1 \end{array} \tag{9.7}$$

Which corresponds to $(25)_{\text{hex}}$. Since the input state as stated before presents only the hexadecimal byte 25, the output state will be C=(25,25,...,25).

9.2.5. Key addition layer

The two inputs to this layer are the current 16-byte state matrix and a sub-key of 16 bytes. The two are combined in a XOR operation, which in the *Galois Field* $GF(2^8)$ is equal to addition.

9.2.6. Key schedule

This step takes the original input key and derives the sub-keys, as in Figure 16 [p. 38]. The number of keys is equal to the number of rounds plus one (due to the key whitening). The AES key schedule is word oriented (1 word = 32 bits).

Example: First round of the key schedule with $K_{0,...,15} = (FF)_{\text{hex}}$.

The first word $W_0 = (FF, FF, FF, FF)$ is XORed with $g(W_3)$.

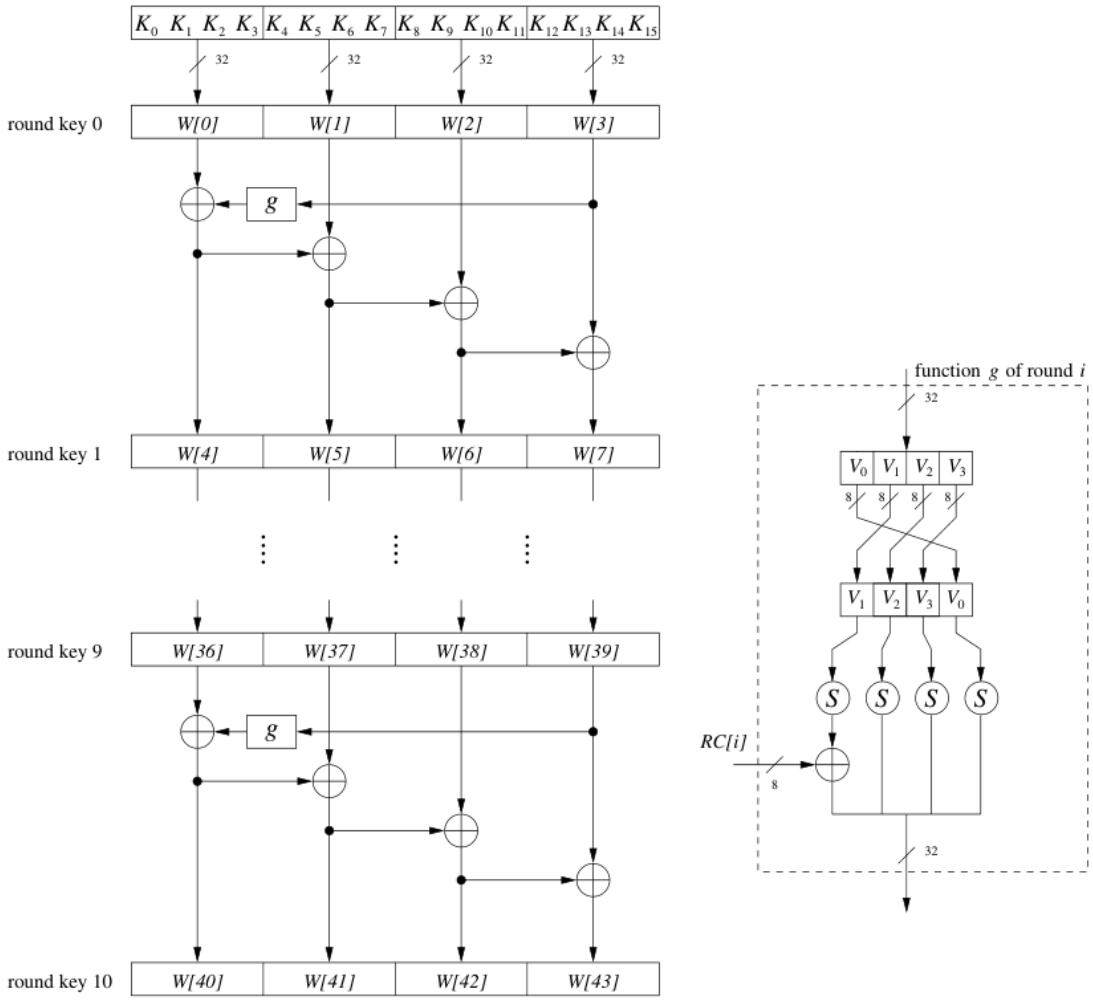


Figure 16: AES key schedule for 128-bit key

The g-box, as shown in Figure 16 on the right, takes the input word and shifts its bytes, then each byte enters in a *S-box*. In our case the bytes are all the same and this transformation gives (see Figure 15 [p. 35]):

$$S(FF) = (16)_{\text{hex}} \tag{9.8}$$

Then the output of the first *S-box* is XORed with $RC[i]$, in this case $RC[1]$. The *round coefficients* RC vary from round to round according to the following rule (computed in $GF(2^8)$, with the same irreducible polynomial, see Section 9.2.3. [p. 34]):

$$\begin{aligned} RC[1] &= x^0 = (00000001)_2 \\ RC[2] &= x^1 = (00000010)_2 \\ &\vdots \\ RC[10] &= x^9 = (00110110)_2 \end{aligned} \tag{9.9}$$

So $(00010110) \oplus (00000001) = (00010111) = (17)_{\text{hex}}$. The output bytes of the g-box will be $(17, 16, 16, 16)_{\text{hex}}$.

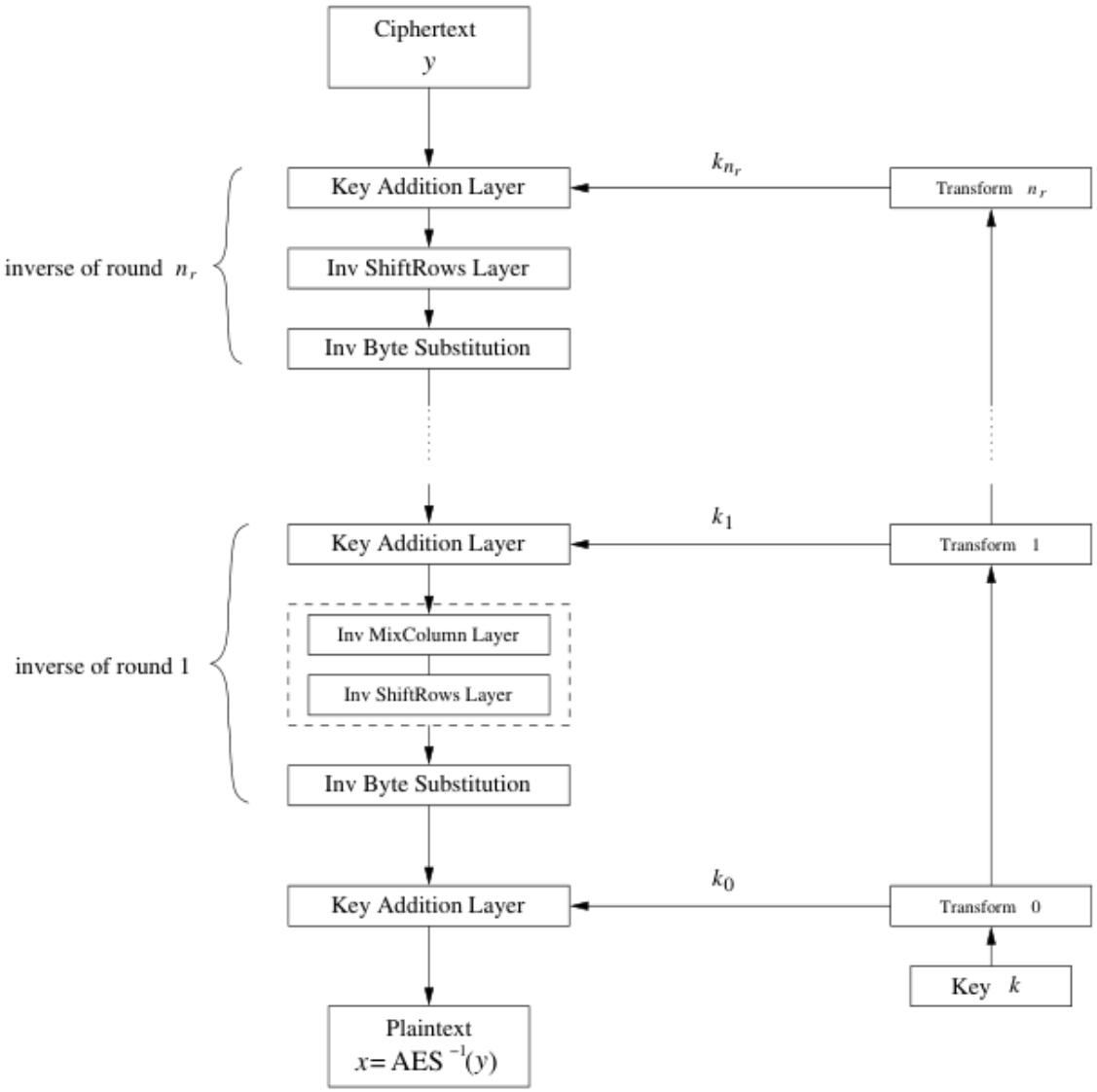
K_1 will then be, after all the XORing, with the bytes in hexadecimal notation:

E8	E9	E9	E9	17	16	16	16	E8	E9	E9	E9	17	16	16	16

Since FF $\oplus$ 16 = E9 and FF $\oplus$ 17 = E8

9.3. Decryption

For what concerns the decryption, all the layers must be inverted and the order of the subkeys is reversed, i.e. we need a reversed key schedule.



9.3.1. Inverse MixColumn Sublayer

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} \tag{9.10}$$

The other columns are computed the same way by multiplying by this matrix. Also here all the elements $\in GF(2^m)$ and the values in the matrix are in hexadecimal notation.

9.3.2. Inverse ShiftRows Sublayer

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

Table 7: Input state

B_0	B_4	B_8	B_{12}	← No shift
B_{13}	B_1	B_5	B_9	← One position shift right
B_{10}	B_{14}	B_2	B_6	← Two positions shift right
B_7	B_{11}	B_{15}	B_3	← Three positions shift right

Table 8: Output state

9.3.3. Inverse byte substitution layer

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Figure 18: Inverse AES S-box

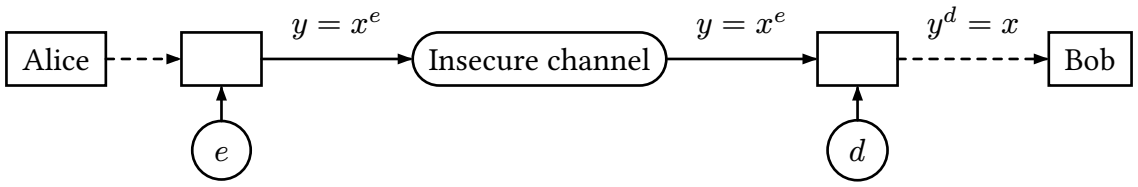
WARNING: The substitution values in the table are the substitution values in AES, comprehensive of the $GF(2^8)$ inverse and the affine mapping! The inverse affine mapping is still reported below for completeness.

$$\begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} \equiv \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \mod 2 \tag{9.11}$$

9.3.4. Decryption key schedule

Since the first decryption round requires the last subkey, the second round requires the second-to-last subkey and so on we need the reversed key schedule, which in practice translates to compute and store all the subkeys and use them as needed.

10. RSA



Advantage: no need of a secure channel

How can it be safe if we are only doing an exponentiation?

Finding the inverse is easy in $\mathbb{R}$, not so difficult in $\mathbb{Z}$ and extremely difficult in $\mathbb{Z}_m$ without knowing m.

10.1. Example

a) Bob choses two prime numbers $p = 5, q = 7$

b) Bob multiplies them and gets $n = p \cdot q = 35$

c) Bob computes $\Phi(35) =$ "# of coprimes with 35, including 1" $= \{k \geq 1, k < 35 \text{ s.t. } \gcd(k, n) = 1\}$

d) Bob choses $e \in \{1, ..., \Phi(n) = 24\}$ in a way that $\gcd(e, 24) = 1$

e) Bob publishes the public key $= (e, n), 11, 35$ (Φ is secret)

f) Alice encrypts the message $x =$ "x" $= 88 \bmod 35 = 18$:
- Encryption:

$$y = 18^e = 18^{11} \bmod 35 = ... = 2 \bmod 35 \tag{10.1}$$

g) To be able to decrypt, Bob must be able to know d, but $d = e^{-1} \bmod \boldsymbol{\Phi(n)}$

To understand why we take the mod $\boldsymbol{\Phi(n)}$, we should look at Corollary 2.1 [p. 5].

10.1.1. Two possible sources of mistake!

a) Working with elements of $\mathbb{Z}_m$, so in $\text{mod}(m)$, the exponents are to be computed $\text{mod}(\Phi(m))$

b) All of this holds in the hypothesis $\gcd(a, m) = 1$. Then under such hypothesis, if $d = e^{-1} \text{mod}(\Phi(n))$ then, $y^d \equiv (x^e)^d \equiv x^{ed} \equiv x^{1 \bmod(\Phi(n))} \equiv x \bmod(n)$

Theorem 10.1:

$$\begin{gathered} p, q \text{ primes and } n = p \cdot q \\ \forall x \in \mathbb{Z}_m, \forall e \in \mathbb{Z}_{\Phi(n)}, \gcd(e, \Phi(n)) = 1 \\ \Downarrow \\ (x^e)^d \equiv x \bmod n \text{ where } d = e^{-1} \text{ in } \mathbb{Z}_{\Phi_n} \end{gathered} \tag{10.2}$$

Proof: We can split the proof into 2 parts:

1. Suppose that $\gcd(x, n) = 1$, then

$$(x^e)^d \equiv x^{ed} \equiv x^{1+t\Phi(n)} \equiv x \cdot \left(x^{\Phi(n)}\right)^t \equiv x \tag{10.3}$$

2. Suppose now that $\gcd(x, n) > 1$. Since $x \neq n$ (since $x \in \mathbb{Z}_n$), we get that either $x = rp$ or $x = sq$.
   Suppose that $x = rp$:

$$\begin{aligned} x^{t\Phi(n)} &= (rp)^{t\Phi(n)} \\ &= x^{t(p-1)(q-1)} \\ &= \left(x^{\Phi(q)}\right)^{t(p-1)} \\ &= 1^{t(p-1)} \bmod q \\ &= 1 \bmod q \\ &= 1 + uq \end{aligned} \tag{10.4}$$

   Now

$$\begin{aligned} x \cdot x^{t\Phi(n)} &= rp(1 + uq) \\ &= rp + rupq \\ &= x + run \\ &\equiv x \bmod n \\ &= x^{1 \cdot \bmod \Phi(n)} \\ &= x^{1+t\Phi(n)} \\ &\implies x^{ed} = x \bmod n \end{aligned} \tag{10.5}$$

□

Remark: This does not imply the existence of x^{-1}

10.2. Complexity of RSA

We take a number $N = \mathcal{ABCD}$, N is "the number" and n is the number of digits of N (equal to 4 in this example).

$n \simeq \log N$

Definition 10.1 (Polynomial complexity): If a mathematical operation requires a number of steps that grows polynomially with n, it is said to be of **polynomial complexity**.

Definition 10.2 (Exponential complexity): If a mathematical operation requires a number of steps that grows polynomially with N, it is said to be of **exponential complexity**

We now look at the complexity of the different steps that need to be performed in RSA:

1. Key generation
   a. Choice of p and q:
      - Polynomial
   b. Compute $n = p \cdot q$:
      - Polynomial
   c. Compute $\Phi(n)$:
      - **Polynomial with the clever way**
      - **Exponential without knowing p and q**
   d. Selection of the public exponent $e \in \mathbb{Z}_{\Phi(n)}$:
      - Easy
   e. Compute the private key $d = e^{-1} \bmod \Phi(n)$:
      - Brute force: exponential
      - Extended Euclidian Algorithm: polynomial (but it requires to know $\Phi(n)$)

2. Encryption:
   a. $x^e = \underbrace{x \cdot x \cdot x ... \cdot x}_{e \text{ times}}$:
      - Exponential
      - Polynomial (with fast exponentiation)

3. Decryption:
   a. Same as encryption

10.3. Euclidian algorithm

It is used to compute the gcd and it is polynomial since the complexity decreases every 2 steps.

10.3.1. Example

Find $\gcd(8151, 5390)$:

$$\begin{gathered} 8151 = 3 \cdot 11 \cdot 14 \cdot 19 \\ 5390 = 2 \cdot 5 \cdot 7^2 \cdot 11 \\ \Downarrow \\ \gcd(8151, 5390) = 11 \end{gathered} \tag{10.6}$$

With the Euclidian algorithm: we write them as integer division with quotient and rest, recursively in order to find the gcd.

$$
\begin{aligned}
8151 &= q \cdot 5390 + r \\
&= 1 \cdot \underset{\downarrow}{5390} + \underset{\downarrow}{2761} \\
5390 &= \underset{\downarrow}{2761} + \underset{\downarrow}{2629} \\
2761 &= \underset{\downarrow}{2629} + \underset{\downarrow}{132} \\
2629 &= \underset{\downarrow}{132} \cdot 19 + \underset{\downarrow}{121} \\
132 &= \underset{\downarrow}{121} + \underset{\downarrow}{11} \\
121 &= 11 \cdot \underbrace{11}_{\gcd} + 0
\end{aligned}
\tag{10.7}
$$

The algorithm stops when $r = 0$ meaning that we found the gcd.

Theorem 10.2:

$$r_0, r_1 \in \mathbb{N}, r_0 > r_1 \tag{10.8}$$

Then

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) \tag{10.9}$$

where

$$r_0 = q r_1 + r_2 \tag{10.10}$$

10.4. Extended Euclidian Algorithm (EEA)

The goal of this algorithm is to solve the Diophantine equation:

$$\gcd(r_0, r_1) = s r_0 + t r_1 \tag{10.11}$$

in the unknowns s and t (which means that r_0, r_1 are given).

As a byproduct, if we want to find b s.t. $ba = 1 \bmod m$, assuming $\gcd(a, m) = 1$, we can then use

$$1 = \gcd(a, m) = sa + tm \overset{\overbrace{\text{mod}}}{\longrightarrow} 1 = sa \bmod m \tag{10.12}$$

and we finally get

$$b = a^{-1} = s \tag{10.13}$$

Example: Find t and s for $r_0 = 8151, r_1 = 5390$

1	$8151 = 5390 \cdot 1 + 2761$	$2761 = \underbrace{1}\cdot 8151 \underbrace{-1}\cdot 5390$
2	$5390 = 2761 + 2629$	$2629 = 5390 - 2761$ $= \underbrace{-1}\cdot 8151 \underbrace{+2}\cdot 5390$
3	$2761 = 2629 + 132$	$132 = 2761 - 2629$ $= \underbrace{2}\cdot 8151 \underbrace{-3}\cdot 5390$
4	$2629 = 19 \cdot 132 + 121$	$121 = 2629 - 19 \cdot 132$ $= \underbrace{-39}\cdot 8151 \underbrace{+59}\cdot 5390$
5	$132 = 121 + 11$	$11 = 132 - 121$ $= \underbrace{41}\cdot 8151 \underbrace{-62}\cdot 5390$
6	$121 = 11 \cdot 11 + 0$	

So $11 = \underbrace{41}_{s} \cdot 8151 \underbrace{-62}_{t} \cdot 5390$.

10.4.1. Euclid-Wallis algorithm

Another way of performing the same calculation is by using the Euclid-Wallis algorithm[1]:

r_0	$B_0 = 0$	$C_0 = 1$	/
r_1	$B_1 = 1$	$C_1 = 0$	/
$r_2 = r_0 - r_1 \cdot D_2$	$B_2 = B_0 - B_1 \cdot D_2$	$C_2 = C_0 - C_1 \cdot D_2$	$D_2 = \lfloor r_0/r_1 \rfloor$
...			
$r_i = r_{i-2} - r_{i-1} \cdot D_i$	$B_i = B_{i-2} - B_{i-1} \cdot D_i$	$C_i = C_{i-2} - C_{i-1} \cdot D_i$	$D_i = \lfloor r_{i-2}/r_{i-1} \rfloor$
...			
$\gcd(r_0, r_1)$	t	s	...
0	/		...

Applying this method to the example above gives:

8151	0	1	
5390	1	0	
2761	$-1 = 0 - (1 \cdot 1)$	$1 = 1 - (0 \cdot 1)$	$1 = \lfloor 8151/5390 \rfloor$
$2629 = 8151 - 1 \cdot 2761$	$2 = 1 - (-1 \cdot 1)$	$-1 = 0 - (-1 \cdot 1)$	$1 = \lfloor 5390/2761 \rfloor$
$132 = 2761 - 1 \cdot 2629$	$-3 = -1 - (1 \cdot 2)$	$2 = 1 - (-1 \cdot 1)$	$1 = \lfloor 5390/2761 \rfloor$
$121 = 2629 - 19 \cdot 132$	$59 = 2 - (-3 \cdot 19)$	$-39 = -1 - (2 \cdot 19)$	$19 = \lfloor 2629/132 \rfloor$
$\gcd = 11 = 132 - (121 \cdot 1)$	$t = -62 = -3 - (59 \cdot 1)$	$s = 41 = 2 - (-39 \cdot 1)$	$1 = \lfloor 132/121 \rfloor$
$0 = 121 - 11 \cdot 11$			$11 = \lfloor 121/11 \rfloor$

[1] You can find a detailed explanation here: https://math.stackexchange.com/a/68021°. Notice that the table has been transposed to make the process easier for computing the inverse of polynomials: Exercise 8.2 [p. 32].

Notice that the algorithm stops when we find 0 in the first column. When we stop we can find the values of interest in the second to last row. Notice that if the two starting numbers r_0 and r_1 are coprime with each other, we will not find a 0 in the first column but rather a 1 (which is the gcd); in this case we will find the inverse of $r_1 \bmod r_0$ in the second column, last row.

10.5. Fast exponentiation

Let's say we want to compute x^8:

Naive method:

$$\underbrace{x \xrightarrow{M} x^2 \xrightarrow{M} \dots \xrightarrow{M} x^8}_{7 \text{ multiplications}} \tag{10.14}$$

Smart method:

$$\underbrace{x \xrightarrow{M} x^2 \xrightarrow{x^2 \cdot x^2} x^4 \xrightarrow{x^4 \cdot x^4} x^8}_{3 \text{ steps}} \tag{10.15}$$

We went from 2^{n-1} steps to n.

10.6. Improvement of RSA

Instead of using $\Phi(n) = (p-1)(q-1)$, we replace it with $m = \gcd(p-1, q-1) < \Phi(n)$

Recalling Section 10.1. [p. 41], where we had $p = 7, q = 5$, we get

$$\begin{gathered} p-1 = 6, q-1 = 4 \\ m = 12 < 24 = \Phi(n) \\ e = \{0, 1, \dots, m-1\} \\ \gcd(e, m) = 1 \end{gathered} \tag{10.16}$$

The protocol still works since $\forall x \in \mathbb{Z}_m, x^{ed} = x \bmod n, d = e^{-1} \in \mathbb{Z}_n$ but the advantage is that the exponentiation is faster.

Theorem 10.3 (Little Fermat Theorem): If p is prime, then $\forall a \in \mathbb{Z}$

$$a^{p-1} \equiv 1 \bmod p \tag{10.17}$$

Theorem 10.3 is a particular case of Euler's theorem, as $\Phi(p) = p - 1 \forall p$ prime

Proof: Consider the set, given $a \in \mathbb{Z}_p$:

$$\begin{gathered} a(\mathbb{Z}_p \setminus \{0\}) = \{a, 2a, 3a, \dots, (p-1)a\} \\ \text{and we want to prove that } \; = \mathbb{Z}_p \setminus \{0\} \end{gathered} \tag{10.18}$$

First we want to prove that $0 \notin a(\mathbb{Z}_p \setminus \{0\})$.
Suppose that $0 \in a(\mathbb{Z}_p \setminus \{0\}) \Leftrightarrow ka = 0 \bmod p$. Now we consider the possible identity of the two elements in $a(\mathbb{Z}_p \setminus \{0\})$, which means the following:

$$ra = sa \quad r, s \in \mathbb{Z}_p \setminus \{0\} \tag{10.19}$$

If p is prime this means that the inverse exists, so that we can multiply both sides of Equation (10.19) by a^{-1}, giving us:

$$r\cancel{aa^{-1}} = s\cancel{aa^{-1}} \tag{10.20}$$

which means that the two elements must coincide. Then, $a(\mathbb{Z}_p \setminus \{0\})$ is made of $p-1$ **distinct** elements of $\mathbb{Z}_p \setminus \{0\}$, which has cardinality equal to $p-1$. Then:

$$\begin{gathered} a(\mathbb{Z}_p \setminus \{0\}) = \mathbb{Z}_p \setminus \{0\} \\ \Downarrow \\ \prod_{k \in \mathbb{Z}_p \setminus \{0\}} k = \prod_{j \in \mathbb{Z}_p \setminus \{0\}} j \\ \cancel{1 \cdot 2 \cdot \ldots \cdot (p-1)} \equiv a \cdot 2a \cdot 3a \cdot \ldots p \dot{-} 1a = \cancel{1 \cdot 2 \cdot \ldots (p-1)} a^{p-1} \\ a^{p-1} \equiv 1 \bmod p \end{gathered} \tag{10.21}$$

□

Theorem 10.4 (RSA with slight improvement): Given $x \in \mathbb{Z}_n$

$$\begin{gathered} x^{ed} = x \bmod n \\ ed = 1 \bmod m \Rightarrow ed = 1 + km \\ x^{ed} = x \cdot x^{km} \end{gathered} \tag{10.22}$$

but $m = r(p-1)$ so:

$$x \cdot x^{kr(p-1)} = x \cdot \left(x^{kr}\right)^{p-1} \equiv x \bmod p \tag{10.23}$$

Proof:

$$\begin{gathered} x^{ed} = 1 \bmod n \Rightarrow ed = km + 1, \\ m = r(p-1) \text{ (since } m \text{ must be a multiple of } (p-1)) \\ x^{ed} = x \cdot x^{km} = x \cdot x^{kr(p-1)} = x\left(x^{kr}\right)^{p-1} \overset{\text{FLT}}{=} x \bmod p \\ \Downarrow \\ p \mid x^{ed} - x \end{gathered} \tag{10.24}$$

Now we can do the same considering $m = s(q-1)$:

$$\begin{aligned} &\begin{cases} q \mid x^{ed} - x \\ p \mid x^{ed} - x \end{cases} \implies \underbrace{pq}_{n} \mid x^{ed} - x \\ &\implies x^{ed} - x = \ell n \\ &\implies x^{ed} - x \equiv 0 \bmod n \\ &\implies x^{ed} = x \bmod n \end{aligned} \tag{10.25}$$

□

Exercise 10.1:

Find all the inverses of the elements in $\mathbb{Z}_{24}$

x	x^{-1}
1	1
2	/
3	/
4	/
5	5
6	/

x	x^{-1}
7	7
11	11
13	13
17	17
19	19
23	23

Every item with an inverse is the inverse of itself like in Liguria
(Cit.)

Why is it true that $\forall x \in \mathbb{Z}_{24}\ x \equiv x^{-1}$ if $\gcd(x, 24) = 1$?

The condition $x^2 = 1$ means $x^2 = 1 + 24k$ for some $k \Leftrightarrow (x-1)(x+1) \cdot 24k$ it is a multiple of 24 thus contains the factors $2^3 \cdot 3$

One among $x-1, x, x+1$ is divided by 3, but not x which is coprime with 24. then either $x-1$ or $x+1$ is divisible by 3 meaning that $(x-1)(x+1) = x^2 - 1$ is divisible by 3.

Moreover x is odd, so $x-1$ and $x+1$ are two consecutive even numbers, then one of them must be divisible by 4. So we have 2^2 from one and 2 from the other and the condition $x^2 = 1 + 24k$ for some k is satisfied $\forall x$ coprime with 24.

For which other $\mathbb{Z}_n$ does it happen the same? $(x+1)(x-1) = kn$. From three consecutive numbers one always collects the factors $2^3 \cdot 3$, then $\frac{n}{24}$ is the necessary and sufficient condition for having $x^2 = 1 \quad \forall x$ coprime with n.

Exercise 10.2: Find the inverse of 19 in $\mathbb{Z}_{999}$.

Since $\gcd(19, 999) = 1$ we can be sure that the inverse t exists and is unique:

$$\begin{gathered}\exists!\ t, s \in \mathbb{Z} \text{ s.t. } 1 = t \cdot 19 + s \cdot 999 \\ \Downarrow \\ 1 \equiv t \cdot 19 \bmod 999 \iff t = 19^{-1}\end{gathered} \tag{10.26}$$

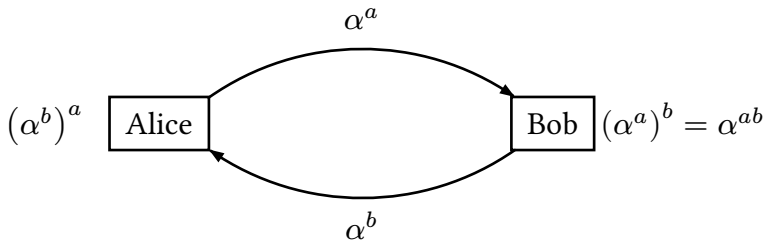
	Euclidian algorithm	Finding the inverse
1	$999 = 52 \cdot 19 + 11$	$11 = \underbrace{1} \cdot 999 \ \underbrace{-52} \cdot 19$
2	$19 = 11 + 8$	$8 = 19 - 11$ $= \underbrace{-1} \cdot 999 \ \underbrace{+53} \cdot 19$
3	$11 = 8 + 3$	$3 = 11 - 8$ $= \underbrace{2} \cdot 999 \ \underbrace{-105} \cdot 19$
We can avoid computing the factors of 999		
4	$8 = 2 \cdot 3 + 2$	$2 = \ldots \ \underbrace{+263} \cdot 19$
5	$3 = 2 + 1$	$1 = \ldots \ \underbrace{-368} \cdot 19$
6	$2 = 2 \cdot 1 + 0$	

So $-368 \equiv 19^{-1}$, which we can also write as $631 \equiv 19^{-1}$ since $631 = 999 - 368$.

This algorithm can be used also for computing multiplicative inverses in Galois fields (Section 8. [p. 29], Exercise 8.2 [p. 32]): In $GF(2^m)$ the inputs of the algorithm are the field element $A(x)$ and the irreducible polynomial.

11. Diffie Hellman

The basic idea is that Alice and Bob must perform 2 mathematical operations that are easy to do and difficult to undo and must **commute** with each other. The difficult property to achieve is the second one: we want an operation easy to do and difficult to undo.



Note: α was sent in the insecure channel and so we must assume that Eve knows that; at this point we can simplify the protocol and make α public (i.e. Alice's public key).

- We assume that Eve knows $\alpha, \alpha^a, \alpha^b$ but she doesn't know neither a nor b.

To find a and b, Eve must solve $\alpha^x = M$, where M is the intercepted message. This is easy to do in $\mathbb{R}$ but not in $\mathbb{Z}_m$.

Definition 11.1 (Discrete logarithm):

$$\begin{gathered} x = \log_\alpha(M) \\ \alpha, x, M \text{ are integers} \end{gathered} \tag{11.1}$$

The discrete logarithm problem is **exponential** in the number of bits.

The exponentiation in $\mathbb{Z}_m$ is easy to do and difficult to undo and is thus a **1-way function**.

11.1. Private and public keys

Note that m must be public too and thus the public key is given by $k_{\text{pub}} = (\alpha, m)$, the private keys, on the other hand, are $k^A_{\text{priv}} = a$ and $k^B_{\text{priv}} = b$

Example: Solve $3^x = 2 \bmod 5$

We can do this by bruteforce and find $x = 3$

Example: Solve $3^x = 7 \bmod 11$

x	0	1	2	3	4	5	6	7	8	9	10
3^x	1	3	9	5	4	1	3	... periodic ...			

This equation has no solution, $\alpha = 3$ and $m = 11$ is a bad public key because it restricts the space of the final key that can be constructed.

Example: Solve $2^x = 7 \bmod 11$

x	0	1	2	3	4	5	6	7	8	9	10
2^x	1	2	4	8	5	10	9	7	3	6	1

In this case we cover all $\mathbb{Z}_{11}$ except 0.

11.1.1. Procedure for key exchange

- Setup (Alice)
  1. Choose a large prime number p ($\mathbb{Z}_p$)
  2. Choose $\alpha \in \{2, ..., p-2\}$
  3. Publish p and α

  Comment: why not $\alpha = p - 1$?

  Because when exponentiating

$$\alpha^a = (p-1)^a = \sum_{j=0}^{a} p^j (-1)^{a-j} \equiv (-1)^{a-j} \tag{11.2}$$

- Exchange
  1. Alice (A) chooses $a = k_{pr,A}$ (private) $\in \{2, ..., p-2\}$
  2. A computes $A = \alpha^a$
  3. A sends α^a to Bob (B)
  4. Analogously, Bob chooses $b = k_{pr,b} \in \{2, ..., p-2\}$
  5. B computes $B = \alpha^b$
  6. Bob sends B to Alice
  7. Alice computes $B^a = \left(\alpha^b\right)^a$ and Bob computes $A^a = \left(\alpha^a\right)^b$

11.2. Finite group theory

We are given a set G with an internal operation $\circ$ (aka *pallicchero*).

Definition 11.2 (Internal operation): An internal operation is a map

$$\left.\begin{array}{l} \mathcal{G} \times \mathcal{G} \mapsto \mathcal{G} \\ (x, y) \mapsto x \circ y \end{array}\right\} \text{This implies that } \mathcal{G} \text{ is closed with respect to " } \circ \text{ "} \tag{11.3}$$

Definition 11.3 (Group): $(\mathcal{G}, \circ)$ is a group if:

1. $\forall x, y, z \in \mathcal{G} \rightarrow (x \circ y) \circ z = x \circ (y \circ z) = x \circ y \circ z$ **(associativity)**
2. $\exists e \in \mathcal{G}$ s.t. $e \circ x = x \circ e = x \quad \forall x \in \mathcal{G}$ **(neutral element)**
3. $\forall x \in \mathcal{G} \quad \exists x^{-1} \in \mathcal{G}$ s.t. $x \circ x^{-1} = x^{-1} \circ x = e$ **(inverse)**

Definition 11.4 (Commutative or abelian group): If a group $(\mathcal{G}, \circ)$ enjoys commutativity, i.e. $\forall x, y \in \mathcal{G} \quad x \circ y = y \circ x$ it is called commutative or abelian.

Does $\mathbb{Z}_m$ form a group?

1. $(\mathbb{Z}_m, +)$? Yes
2. $(\mathbb{Z}_m, \circ)$? No, because "0" never has an inverse
3. $(\mathbb{Z}_m \setminus \{0\}, \circ)$ **Yes, but only if m is prime**

Definition 11.5: Given $m \in \mathbb{N}$, we can define

$$\mathbb{Z}_m^* = \mathbb{Z}_m \setminus \{n \in \mathbb{Z}_m \text{ s.t.} \gcd(n, m) > 1\} \tag{11.4}$$

It is easy to show that $\mathbb{Z}_m^*$ is a group, in fact:

1. Associativity comes from $\mathbb{Z}_m$
2. The neutral element "1" is in $\mathbb{Z}_m^*$ because $\gcd(1, m) = 1$
3. The inverse exists by definition of $\mathbb{Z}_m^*$
4. $\mathbb{Z}_m^*$ is closed with respect to multiplication:

$$x \cdot y \in \mathbb{Z}_m^* \quad \gcd(x \cdot y, m) = 1 \tag{11.5}$$

because neither x nor y have a common factor with m.

We can notice that $\#\mathbb{Z}_m^* = |\mathbb{Z}_m^*| = \Phi(m)$.

Definition 11.6 (Cyclicity): $(\mathcal{G}, \circ)$ is **cyclic** if $\exists\ \alpha \in \mathcal{G}$ s.t.

$$\text{Gen}(\alpha) \triangleq \{\alpha^n, \quad n = 1, ..., |\mathcal{G}|\} = \mathcal{G} \tag{11.6}$$

with

$$\alpha^n = \underbrace{\alpha \circ \alpha \circ ... \circ \alpha}_{n \text{ times}} \tag{11.7}$$

Definition 11.7 (Generator): α is said **generator** of $\mathcal{G}$ and also a **primitive element**.

Definition 11.8 (Subgroup): A subgroup $(\mathcal{S}, \circ)$ of a group $(\mathcal{G}, \circ)$ is a subset of $\mathcal{G}$ that has the structure of a group.

A subset $\mathcal{S}$ of a group $(\mathcal{G}, \circ)$ is a subgroup of $\mathcal{G}$ if and only if $\mathcal{S}$ is closed with respect to "$\circ$".

Remark: Given a group $(\mathcal{G}, \circ)$, the subset of $\mathcal{G}$ defined by $\text{Gen}(\beta) = \{\beta^n, n = 1, ..., |G|\}$ is a subgroup if $(\mathcal{G}, \circ)$.

Indeed, if we consider two elements of $\text{Gen}(\beta) : x$ and y, it must be

$$\begin{gathered} x = \beta^{n_1} \wedge y = \beta^{n_2} \\ \Downarrow \\ x \cdot y = \beta^{n_1 + n_2} = \beta^{|\mathcal{G}|} \cdot \beta^{n_1 + n_2 \bmod |\mathcal{G}|} \end{gathered} \tag{11.8}$$

If $\mathcal{G} = \mathbb{Z}_m^*$, then, by Euler's theorem (Theorem 2.5 [p. 5]) we get that $\beta^{|\mathcal{G}|} = 1 \bmod m$ and finally

$$xy = \beta^{n_1+n_2 \bmod |\mathcal{G}|} \in \text{Gen}(\beta) \tag{11.9}$$

Definition 11.9 (Order of an element of a group): Given $(\mathcal{G}, \circ)$ and $\alpha \in \mathcal{G}$

$$\text{ord}(\alpha) = \min(n, \text{s.t. } \alpha^n = \alpha) \tag{11.10}$$

(which is to say it is the "period of the orbit of α")

Definition 11.10 (Order or cardinality of a group): A group $(G, \circ)$ is finite if it has a finite number of elements. We denote the **cardinality** or **order** of the group G by $|G|$.

Theorem 11.1 (Generalization of Euler's): Let $(\mathcal{G}, \circ)$ be a finite group (i.e. with a finite number of elements), then $\forall \alpha \in \mathcal{G}$:
1. $\alpha^{|\mathcal{G}|} = e$ (where e is the neutral element)
2. $\text{Ord}(\alpha)$ divides $|\mathcal{G}|$

Proof: (For $\mathbb{Z}_m^*$)

$|\mathcal{G}| = \Phi(m)$, then

$$\begin{aligned} & & \alpha^{|\mathcal{G}|} &= \alpha^{\Phi(m)} \\ \text{Euler, since } \gcd(\alpha, m) = 1 & \longrightarrow & &\equiv 1 (\bmod\, m) \end{aligned} \tag{11.11}$$

□

Theorem 11.2: For every prime p, $\left(\mathbb{Z}_p^*, \circ\right)$ is an abelian, finite and cyclic group.

This is important for Diffie Hellman since we want to have m prime and chose α as a generator of $\mathbb{Z}_m^*$

Theorem 11.3: Let $(\mathcal{G}, \circ)$ be a finite cyclic group, then:

1. The number of generators of $\mathcal{G}$ is $\Phi(|\mathcal{G}|)$
2. If $|\mathcal{G}|$ is prime, then all elements $\alpha \neq e$ are generators.

Exercise 11.1: Determine wether the following groups are cyclic and, if so, find a generator:

1. $\mathbb{Z}_3^*$:
   $\mathbb{Z}_3^* = \{1, 2\} \quad 2^1 = 2, 2^2 = 4 \equiv 1 \implies$ cyclic and 2 is a generator.
2. $\mathbb{Z}_6^*$:
   $\mathbb{Z}_6^* = \{1, 5\} \quad 5^1 = 5, 5^2 = 1 \implies$ cyclic and 5 is a generator
3. $\mathbb{Z}_8^*$:
   $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$

$$\begin{aligned} &3^1 = 3 \quad 3^2 = 1 \quad 3^3 = 3 \implies 3 \text{ is not a generator} \\ &7^1 = 7 \quad 7^2 = 1 \qquad\quad \implies 7 \text{ is not a generator} \\ &5^1 = 5 \quad 5^2 = 1 \qquad\quad \implies 5 \text{ is not a generator} \\ &\qquad\qquad\qquad\quad \Downarrow \\ &\qquad\qquad\qquad\quad \mathbb{Z}_8^* \text{ not cyclic} \end{aligned} \tag{11.12}$$

11.3. Notes about Diffie Hellman

- $|\mathbb{Z}_p^*| = p - 1$ is **not prime**! So the number of generators of $\mathbb{Z}_p^*$ is $\Phi(p-1)$, then the choice of an optimal α for Diffie Hellman is non trivial.
- Why don't we chose $\alpha = p - 1$? Because when we exponentiate we get the following

$$\alpha^a = (p-1)^a = \sum_j^a p^j (-1)^{a-j} \tag{11.13}$$

  wich gives either 1 or $p - 1$.

- Why don't we chose a and/or b equal to $p - 1$? Because, for Fermat's little theorem (Theorem 10.3 [p. 46])

$$\alpha^a = \alpha^{p-1} \equiv 1 \tag{11.14}$$

- It is important that α generates a large subgroup of $\mathbb{Z}_p^*$.

Theorem 11.4: Let $(\mathcal{G}, \circ)$ a finite cyclic group of cardinality n and let α be a generator of $\mathcal{G}$.

$\forall k$ that divides n, $\exists!$ cyclic subgroup $\mathcal{H}$ of $\mathcal{G}$ which has k elements.

This subgroup is generated by $\alpha^{n/k}$ and consists of all elements a s.t. $a^k = 1$.

There are no other subgroups.

Example: In the case of $\mathbb{Z}_{11}^*$, $\alpha = 8$ is a primitive (the order of the element $8 = 10$, thus it generates all the elements of the group). In order to find a generator of a subgroup β, for example the one of order 2, we apply the theorem above, so:

$$\beta = \alpha^{\frac{n}{k}} = 8^{\frac{10}{2}} = 10 \bmod 11 \tag{11.15}$$

Where n is the order (cardinality) of the group, k is the order (cardinality) of the subgroup.

11.4. How to find a generator

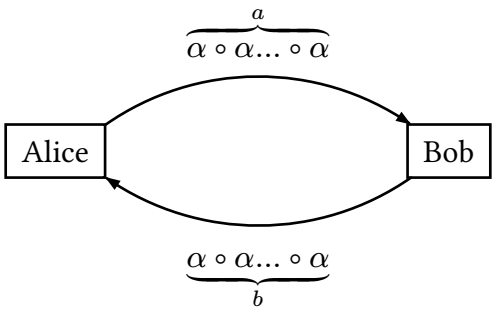
Given $\mathbb{Z}_p^*$, suppose it is a cycle, then how do we find a generator?

This is a difficult problem and, even if we manage to do that, we can use the Pohlig–Hellman algorithm to find the key in sub-exponential time.

We often look for generators of subgroups of $\mathbb{Z}_p^*$ of cardinality k, with k prime.

We can generalize the procedure seen above (Section 11.1.1. [p. 51]) to overcome these problems:

We chose a group $(\mathcal{G}, \circ)$ and $\alpha \in \mathcal{G}$. Both $\mathcal{G}$ and α are public.



So, what Bob does is the following

$$\underbrace{\overbrace{\alpha \circ \alpha ... \circ \alpha}^{a \text{ times}} \circ \overbrace{\alpha \circ \alpha ... \circ \alpha}^{a \text{ times}} \circ ... \circ \overbrace{\alpha \circ \alpha ... \circ \alpha}^{a \text{ times}}}_{b \text{ times}} \tag{11.16}$$

while Alice performs the analogous operation with a and b swapped:

$$\underbrace{\overbrace{\alpha \circ \alpha ... \circ \alpha}^{b \text{ times}} \circ \overbrace{\alpha \circ \alpha ... \circ \alpha}^{b \text{ times}} \circ ... \circ \overbrace{\alpha \circ \alpha ... \circ \alpha}^{b \text{ times}}}_{a \text{ times}} \tag{11.17}$$

By doing this, both Alice and Bob end up with "$\alpha \circ ... \circ \alpha$" repeated "$a \cdot b$" times.

What we would like to do now is choose a group $(\mathcal{G}, \circ)$ and an $\alpha \in \mathcal{G}$ such that the operation of composition is difficult to invert knowing α, β and that

$$\underbrace{\alpha \cdot ... \cdot \alpha}_{x \text{ times}} = \beta \tag{11.18}$$

With these assumptions we would like to find

$$x \triangleq \log_\alpha(\beta) \tag{11.19}$$

Definition 11.11 (Generalized discrete logarithm problem):

$$x \triangleq \log_{\alpha}(\beta) \tag{11.20}$$

Elliptic curves provide a group structure where the discrete logarithm problem is very hard.

Exercise 11.2: 8.1 and 8.3 from "Understanding cryptography"
Determine the order of all elements of the multiplicative groups of:
1. $\mathbb{Z}_5^*$
2. $\mathbb{Z}_7^*$
3. $\mathbb{Z}_{13}^*$

1.

a	1	2	3	4
$\text{ord}(a)$	1	4	4	2

2.

a	1	2	3	4	5	6
$\text{ord}(a)$	1	3	6	3	6	2

3.

a	1	2	3	4	5	6	7	8	9	10	11	12
$\text{ord}(a)$	1	12	3	6	4	12	12	4	3	6	12	2

How many elements does each of the multiplicative groups have?:
1. $|\mathbb{Z}_5^*| = 4$
2. $|\mathbb{Z}_7^*| = 6$
3. $|\mathbb{Z}_{13}^*| = 12$

Do all orders from above divide the number of elements in the corresponding multiplicative group?

Yes.

Which of the elements from the tables are primitive elements?
1. $\mathbb{Z}_5^*$:$2, 3$
2. $\mathbb{Z}_7^*$:$3, 5$
3. $\mathbb{Z}_{13}^*$:$2, 6, 7, 11$

Verify for the groups that the number of primitive elements is given by $\phi\left(\left|\mathbb{Z}_p^*\right|\right)$.
1. $\mathbb{Z}_5^*$:$\phi(4) = 2$
2. $\mathbb{Z}_7^*$:$\phi(6) = 2$
3. $\mathbb{Z}_{13}^*$:$\phi(12) = 4$

Exercise 11.3: 8.5 from "Understanding cryptography"
Compute the two public keys and the common key for the DHKE scheme with the parameters $p = 467, \alpha = 2$:

1. $a = 3, b = 5$
2. $a = 400, b = 134$
3. $a = 228, b = 57$

1. $a = 3, b = 5$:

$$k_A = 2^3 = 8,\ k_B = 2^5 = 32,\ k_{AB} = 2^{3*5} = 78 \bmod 467 \tag{11.21}$$

2. $a = 400, b = 134$:

$$\begin{gathered} k_A = 2^{400} = 137 \bmod 467,\ k_B = 2^{134} = 84 \bmod 467, \\ k_{AB} = 2^{400 \cdot 134} = 90 \bmod 467 \end{gathered} \tag{11.22}$$

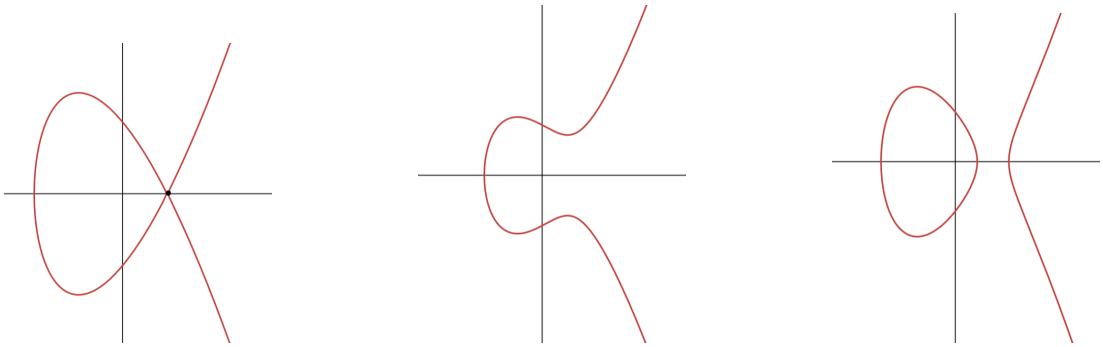
3. $a = 228, b = 57$:

$$\begin{gathered} k_A = 2^{228} = 394 \bmod 467,\ k_B = 2^{57} = 313 \bmod 467, \\ k_{AB} = 2^{228 \cdot 57} = 206 \bmod 467 \end{gathered} \tag{11.23}$$

12. Elliptic curves

Definition 12.1 (Elliptic curve in $\mathbb{R}_2$): An elliptic curve in $\mathbb{R}_2$ is the set of points (x, y) that satisfies:

$$\begin{cases} y^2 = x^3 + ax + b \\ 4a^3 + 27b^2 = 0 \end{cases} \quad a, b \in \mathbb{R} \tag{12.1}$$



(a) $4a^3 + 27b^2 = 0$ (b) $4a^3 + 27b^2 > 0$ (c) $4a^3 + 27b^2 < 0$

Remarks

- The upper branch tends to $y \sim x^{3/2}$ as $x \to \infty$
- The curve is symmetric with respect to the x axis, i.e. if $(x, y) \in EC \implies (x, -y) \in EC$

We now want to find a group structure in the elliptic curve, namely an internal operation "+" (improperly called "sum") that takes a couple of points in the EC and maps them to the elliptic curve itself:

$$\begin{aligned} + : EC \times EC &\mapsto EC \\ (x_1, y_1) \times (x_2, y_2) &\mapsto (x_3, y_3) \end{aligned} \tag{12.2}$$

For the following, we will need another point, called **point at infinity** $\mathcal{O}$. We will then work with $\mathcal{G}' = EC \cup \{\mathcal{O}\}$.

Definition 12.2 (Sum):

$$\begin{aligned} P &= (x_1, y_1) \\ Q &= (x_2, y_2) \\ R &= (x_3, y_3) = P + Q \end{aligned} \tag{12.3}$$

1. $\forall P \in \mathcal{G}' \quad P + \mathcal{O} \triangleq P = \mathcal{O} + P$
2. If $x_1 \neq x_2$ we consider the line going through P and Q, it will intersect the EC in another point R'. R is the symmetric of R' with respect to the x axis:

$$\begin{cases} x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right) \cdot (x_1 - x_3) \end{cases} \tag{12.4}$$

3. If $x_1 = x_2$ and $y_1 \neq y_2$ (the points are one above the other), then $P + Q = \mathcal{O} \implies Q = P^{-1}$
4. If $P = Q$ and the tangent to the EC passing through P is not vertical, then it will intercept the EC in another point R'. Similarly to above, the sum will be the symmetric of R' with respect to the x axis. What we get is

$$\begin{cases} x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \\ y_3 = -y_1 + \frac{3x_1^2 + a}{2y_1}(x_1 - x_3) \end{cases} \tag{12.5}$$

   which is coherent with case 2.
5. If $P = Q$ and the tangent passing through them is vertical then $P + Q = \mathcal{O}$

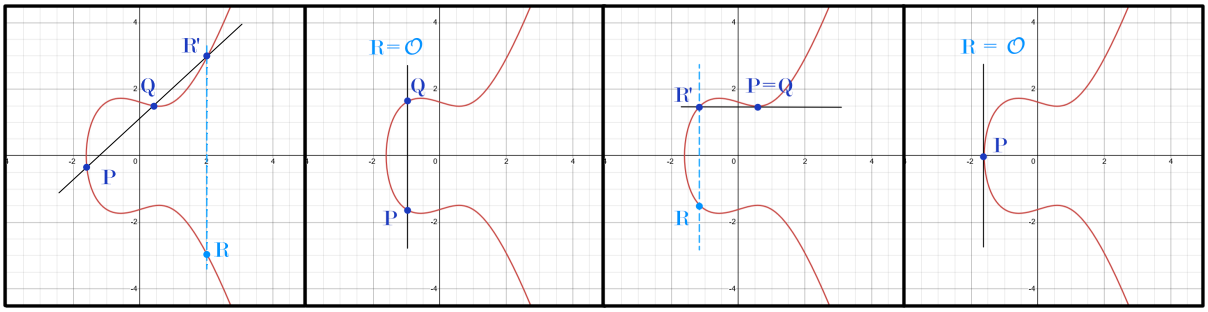


Figure 20: Sum with elliptic curves

Proposition

$(\mathcal{G}', +)$ is an Abelian group.

Since we want to work with finite fields (and elliptic curves are infinite), we work in modulo.

Definition 12.3: Consider $p > 3$ prime. Let $a, b \in \mathbb{Z}_p$ s.t. $4a^3 + 27b^2 \neq 0 \bmod p$

$$\mathcal{G} = \{(x, y) \text{ s.t. } y^2 = x^3 + ax + b \bmod p\} \cup \{\mathcal{O}\} \tag{12.6}$$

with the internal operation given by the formulas for $\mathcal{G}'$ (Definition 12.2 [p. 58]) in $\bmod p$.

The following theorem tells us how many points we have in the resulting group $\mathcal{G}$:

Theorem 12.1 (Hasse):

$$p + 1 - \sqrt{p} \leq |\mathcal{G}| \leq p + 1 + 2\sqrt{p} \tag{12.7}$$

13. Error correcting codes

In communication, errors always occur and we want to be able to **detect** and **correct** them.

Definition 13.1 (Alphabet): Let A a finite set of symbols called an **alphabet**

Definition 13.2 (n-word):

$$A^n = \underbrace{A \times A \times ... \times A}_{n \text{ times}} = \left\{(a^1, ..., a^n), a_j \in A\right\} \tag{13.1}$$

is the set of words of n characters

Definition 13.3 (Code): Fixed A and n, a set $C \subseteq A^n$ is called **code**

13.1. Hamming distance

Definition 13.4 (Hamming distance): The map

$$\begin{aligned} d_H : A^n \times A^n &\mapsto \mathbb{R}^+ \\ (\vec{x}, \vec{y}) &\mapsto d_H(\vec{x}, \vec{y}) = \#\left\{i \text{ s.t. } x^i \neq y^i\right\} \end{aligned} \tag{13.2}$$

is said **Hamming distance** between two n-words.

13.1.1. Properties

1. $d_H(\vec{x}, \vec{y}) = 0$ iff $\vec{x} = \vec{y}$
2. $d_H(\vec{x}, \vec{y}) = d_H(\vec{y}, \vec{x})\ \forall \vec{x}, \vec{y}$
3. $d_H(\vec{x}, \vec{y}) \leq d_H(\vec{x}, \vec{z}) + d_H(\vec{y}, \vec{z})\ \forall \vec{x}, \vec{y}, \vec{z}$ (triangular inequality)

The Hamming distance could give us an idea for implementing error correction:

when we receive $\vec{r} \notin C$, we find $\vec{x}$ s.t. $d_H(\vec{r}, \vec{x}) = \min_{\vec{y} \in C}(\vec{r}, \vec{y})$. The biggest problem with this idea is that the min may not be unique.

Another possible idea is based on supposing that the number of possible error is 1. If the code C is constructed so that the minimum distance between two elements is 3, then $\vec{x}$ (the original word) is the only minimizer in C of the distance from $\vec{x}$.

Definition 13.5 (Minimal distance): Given A, n, C, the **minimal distance** of the code is

$$d(C) = \min_{\substack{\vec{x}, \vec{y} \in C \\ \vec{x} \neq \vec{y}}} d_H(\vec{x}, \vec{y}) \tag{13.3}$$

Theorem 13.1 (Detection of the error): Let $C \subseteq A^n$ with minimum distance d, then C detects $d-1$ errors.

Theorem 13.2 (Correction of the error): Let $C \subseteq A^n$ with minimum distance d, then C corrects

$$e = \left\lfloor \frac{d-1}{2} \right\rfloor \tag{13.4}$$

errors.

Exercise 13.1: We have a channel that applies the X gate with a probability $q = 1 - p$.

Alice sends 2 pairs with the following probabilities:

State	Probability
$\|\varphi^+\rangle\|\varphi^+\rangle = \|\alpha\rangle$	p^2
$\|\Theta^+\rangle\|\varphi^+\rangle = \|\beta\rangle$	pq
$\|\varphi^+\rangle\|\Theta\rangle = \|\gamma\rangle$	pq
$\|\Theta\rangle\|\Theta\rangle = \|\delta\rangle$	q^2

where $|\Theta\rangle$ is the state of the pair after corruption.

1. Rewrite the state in the form $|A_1 A_2 B_1 B_2\rangle$:

$$\begin{aligned}
2|\alpha\rangle &= \left(\left|0^{A_1}0^{B_1}\right\rangle + \left|1^{A_1}1^{B_1}\right\rangle\right)\left(\left|0^{A_2}0^{B_2}\right\rangle + \left|1^{A_2}1^{B_2}\right\rangle\right) \\
&= |0000\rangle + |0101\rangle + |1010\rangle + |1111\rangle \\
2|\beta\rangle &= (|00\rangle + |11\rangle)(|01\rangle + |10\rangle) \\
&= |0001\rangle + |0101\rangle + |1011\rangle + |1110\rangle \\
2|\gamma\rangle &= (|01\rangle + |10\rangle)(|00\rangle + |11\rangle) \\
&= |0010\rangle + |0111\rangle + |1000\rangle + |1101\rangle \\
2|\delta\rangle &= (|01\rangle + |10\rangle)(|01\rangle + |10\rangle) \\
&= |0011\rangle + |0110\rangle + |1001\rangle + |1100\rangle
\end{aligned} \tag{13.5}$$

2. Both Alice and Bob perform the unitary transformation U, where the ordering of the basis vectors is $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. What happens to the states?

$$\begin{gathered} U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\ |\alpha\rangle \text{ remains as is (3 and 4 swap)} \\ |\beta\rangle \text{ remains as is (3 and 4 swap)} \\ U|\gamma\rangle = |\delta\rangle \\ U|\delta\rangle = |\gamma\rangle \\ \text{Notice that the probabilities for } |\gamma\rangle \text{ and } |\delta\rangle \text{ are now swapped} \end{gathered} \tag{13.6}$$

3. Alice and Bob now measure their second qubit in the computational basis. If their result is the same, they keep the first qubits, otherwise they discard them. What is the probability that the first qubit pair is in the state $|\varphi^+\rangle$ **given that** they found the same result in the measurement?

Case $|\alpha\rangle$:

$$\begin{aligned} 2|\alpha\rangle = |0\rangle_{A_1}|0\rangle_{A_2}|0\rangle_{B_1}|0\rangle_{B_2} + |0\rangle_{A_1}|1\rangle_{A_2}|0\rangle_{B_1}|1\rangle_{B_2} + \\ +|1\rangle_{A_1}|0\rangle_{A_2}|1\rangle_{B_1}|0\rangle_{B_2} + |1\rangle_{A_1}|1\rangle_{A_2}|1\rangle_{B_1}|1\rangle_{B_2} \end{aligned} \tag{13.7}$$

Suppose A measures A_2 and finds 0:

$$\begin{aligned} 2|\alpha\rangle = \big(|0\rangle_{A_1}|0\rangle_{B_1}|0\rangle_{B_2} + |1\rangle_{A_1}|0\rangle_{B_1}|0\rangle_{B_2}\big)|0\rangle_{A_2} + \\ \big(|0\rangle_{A_1}|0\rangle_{B_1}|1\rangle_{B_2} + |1\rangle_{A_1}|1\rangle_{B_1}|1\rangle_{B_2}\big)|1\rangle_{A_2} \\ \text{which collpses in} \to \big(|0\rangle_{A_1}|0\rangle_{B_1}|0\rangle_{B_2} + |1\rangle_{A_1}|1\rangle_{B_1}|0\rangle_{B_2}\big)|0\rangle_{A_2} \end{aligned} \tag{13.8}$$

Now, as Bob measures he finds 0:

$$\to \big(|0\rangle_{A_1}|0\rangle_{B_1} + |1\rangle_{A_1}|1\rangle_{B_1}\big)|0\rangle_{B_2}|0\rangle_{A_2} \propto \Phi^+_{(A_1,B_1)}|0\rangle_{B_2}|0\rangle_{A_2} \tag{13.9}$$

Analogously if Alice finds 1, after collapse the state is:

$$\propto \Phi^+_{(A_1,B_1)}|1\rangle_{B_2}|1\rangle_{A_2} \tag{13.10}$$

Exercise 13.2: We have a [7,4] binary (i.e. $A = \{0, 1\}$) Hamming code

$$C = \left\{ \begin{matrix} (x_1, ..., x_7) \text{ s.t. } x_1 + x_2 + x_3 + x_5 = 0 \\ \vdots \end{matrix} \right\} \tag{13.11}$$

1. If $\vec{x} \in C$, given $\lambda \in \{0, 1\}$, say if $\lambda\vec{x} \in C$. Yes, $\lambda\vec{x} \in C$ because $0 \in C$ and $\vec{x} \in C$

2. If $\vec{x}, \vec{y} \in C$, does $\vec{x} + \vec{y} \in C$?

$$\begin{aligned} &(x+y)_1 + (x+y)_2 + (x+y)_3 + (x+y)_5 \\ &= x_1 + y_1 + x_2 + y_2 + x_3 + y_3 + x_5 + y_5 \\ &= \underbrace{x_1 + x_2 + x_3 + x_5}_{\tilde{x}} + \underbrace{y_1 + y_2 + y_3 + y_5}_{\tilde{y}} \end{aligned} \tag{13.12}$$

   both $\tilde{x}$ and $\tilde{y} \in C$, this means that the sum of the coefficients $1, 2, 3, 5$ is equal to 0 by Equation (13.11) [p. 63]. Since they are both equal to 0 their sum is in C which means that C is close with respect to the sum.

Since both points (1. and 2.) are satisfied, we can say that C is a vector subspace called a **linear code**. All the operations we made are possible becaues $\mathbb{Z}_2$ has a natural sum and a natural product. In other words, $\mathbb{Z}_2$ is a field

Definition 13.6 (Linear code): Given a finite field $A = F$ so that F^n has a natural structure of vector space. $C \in F^n$ is said to be a **linear code** if it is a linear subspace of F^n. This is equivalent to saying that C is linear if:
1. C is not empty
2. $\forall \vec{x}, \vec{y} \in C \implies (\vec{x} + \vec{y}) \in C$
3. $\forall \alpha \in F, \forall \vec{x} \in C \implies \alpha \vec{x} \in C$

Definition 13.7 ($[n, k]$ linear code): Let F a finite field. A linear code $C \in F^n$ of dimension k is called a $\boldsymbol{[n, k]}$ **linear code**.

Definition 13.8 (Generator matrix): Given a $[n, k]$ linear code C, a **generator matrix** of C is a matrix with k rows and n columns such that the rows form a basis for C.

Note that G generates C in the sense that

$$\vec{x} \in C \iff \exists \vec{\alpha} \in F^k \text{ s.t. } \vec{\alpha} G = \vec{x} \tag{13.13}$$

A generator matrix for C where the $k \times k$ identity matrix appears at the left is called a **standard generator** matrix.

Notice that, in a $[n, k]$ code defined in $\mathbb{Z}_r^n$, the length of each codeword is n, the number of codewords is r^k and the dimension of the code is k.

Example:

$$C = \left\{ (x_1, ..., x_7) \text{ s.t. } \begin{array}{l} x_1 + x_2 + x_3 + x_5 = 0 \\ x_1 + x_2 + x_4 + x_6 = 0 \\ x_2 + x_3 + x_4 + x_7 = 0 \end{array} \right\}$$

$$G = \overbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}}^{x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7} \tag{13.14}$$

Identity matrix

Obtained from the constraints above

G is a generator matrix for the $[7, 4]$ Linear Hamming Code, where the identity matrix is the basis for the first four digits (x_1, x_2, x_3, x_4) while the other columns can be written as $x_5 = x_1 + x_2 + x_3 (\text{mod } 2), x_6 = x_1 + x_2 + x_4 (\text{mod } 2), x_7 = x_2 + x_3 + x_4 (\text{mod } 2)$.

The lines of G are linearly independent.

13.1.2. Computing $d_{\min}$ for a linear code

Definition 13.9 (Weight of an element): The weight $w(\vec{x})$ of an element $\vec{x} \in F^n$ is

$$w(\vec{x}) = \#\{i \text{ s.t. } x_i \neq 0\} \tag{13.15}$$

Theorem 13.3: Given a linear code $C \in F^n$ then

$$w(\vec{x}) = d_H(0, \vec{x}) \tag{13.16}$$

$$\begin{aligned} d_{\min}(C) &= w_{\min}(C) = \\ &= \min\{w(\vec{x}), \vec{x} \in C, \vec{x} \neq 0\} \end{aligned} \tag{13.17}$$

Proof:

$$\begin{aligned} w_{\min}(C) &= \min\{d(0, \vec{x}), \vec{x} \in C, \vec{x} \neq 0\} \\ &\geq \min\{d(\vec{y}, \vec{x}), \vec{y}, \vec{x} \in C, \vec{x} \neq \vec{y}\} \end{aligned} \tag{13.18}$$

since $0 \in C$ due to the linearity of C □

13.2. Check matrix

The check matrix H answers to the question: given $\vec{x} \in F^n$, does $\vec{x}$ belong to C?

The answer is yes if and only if $H\vec{x}^{\mathsf{T}} = 0$, namely if $C = \ker(H)$

Definition 13.10 (Dual code): Given a code $C \in F^n$, the **dual code** of C is defined as

$$C^{\perp} \triangleq \{\vec{x} \in F^n, \vec{x} \cdot \vec{c} = 0\} \forall c \in C \tag{13.19}$$

Note that, in general $C^{\perp\perp} = \text{span}(C)$. In particular, if C is linear, $C^{\perp\perp} = C$.

Furthermore, whatever C (even non linear), $C^{\perp\perp}$ is linear.

Theorem 13.4: If $C \subseteq F^n$ is an $[n, k]$ linear code, then $C^{\perp}$ is an $[n, n-k]$ linear code.

Proposition

$$\vec{x} \in C^{\perp} \iff G\vec{x}^{\mathsf{T}} = 0 \tag{13.20}$$

where G is a generator matrix of C.

Remark:

If H is a generator matrix for $C^{\perp}$, by the previous theorem $\vec{x} \in C^{\perp\perp}$ if and only if $H\vec{x}^{\mathsf{T}} = 0$. Now, if C is linear, then $C^{\perp\perp} = C$ and the generator matrix for $C^{\perp}$ is a check matrix for C.

Example: Find a check matrix for the binary Hamming Code $[7, 4]$. This means that we have to find a generator for $C^{\perp}$. First of all, saying that $\vec{x} \in C^{\perp} \iff G\vec{x}^{\mathsf{T}} = 0$ is the same as stating the following:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} x_1 + x_5 + x_6 \\ x_2 + x_5 + x_6 + x_7 \\ x_3 + x_5 + x_7 \\ x_4 + x_6 + x_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{13.21}$$

$$\Downarrow$$

$$\begin{cases} x_1 + x_5 + x_6 = 0 \\ x_2 + x_5 + x_6 + x_7 = 0 \\ x_3 + x_5 + x_7 = 0 \\ x_4 + x_6 + x_7 = 0 \end{cases} \Rightarrow \begin{cases} x_1 = x_5 + x_6 \\ x_2 = x_5 + x_6 + x_7 \\ x_3 = x_5 + x_7 \\ x_4 = x_6 + x_7 \end{cases}$$

So a basis of $C^{\perp}$ is:

$$\begin{aligned} \vec{y_1} &= (1110100) \\ \vec{y_2} &= (1101010) \\ \vec{y_3} &= (0111001) \end{aligned} \tag{13.22}$$

which leads to the **check matrix** H.

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{13.23}$$

H coincides with the coefficient matrix of the system defining C

13.3. Syndrome

Suppose that Alice sends $\vec{c} \in C$ (linear) and that Bob receives (e.g. due to corruption) $\vec{r}$. Suppose also that the channel is not so bad and Bob is able to detect the error.

Definition 13.11 (Error vector):

$$\vec{e} = \vec{r} - \vec{c} \tag{13.24}$$

Since $H\vec{c} = 0$ for the properties of H and $H\vec{c} = H(\vec{r} - \vec{e}) \implies H(\vec{r}) = H(\vec{e})$, to find the error $\vec{e}$, Bob has to look among the vectors that have the same image of $\vec{r}$ through H.

Definition 13.12 (Syndrome): Let $C \subseteq F^n$ and $[n, k]$ linear code, and let H a check matrix for C. Let $\vec{r} \in F^n$ (but not necessarily in C). Then

$$\vec{s} = H\vec{r}^{\mathsf{T}} \in F^{n-k} \tag{13.25}$$

is called a **syndrome** of $\vec{r}$.

Theorem 13.5: $\vec{x}, \vec{y} \in F^n$ have the same syndrome if and only if

$$\exists\ \vec{c} \in C \text{ s.t. } \vec{y} = \vec{x} + \vec{c} \tag{13.26}$$

Definition 13.13 (Coset): Given a linear code C, we define a **coset** of $\vec{x}$ by:

$$\vec{x} + C \triangleq \{\vec{x} + \vec{c}, \vec{c} \in C\} \tag{13.27}$$

Theorem 13.6: Given a linear code C, then:
i. $\vec{x} \in \vec{y} + C \Longrightarrow \vec{x} + C = \vec{y} + C$
ii. $\forall \vec{x}, \vec{y} \in F^n$, either $\vec{x} + C = \vec{y} + C$ or $(\vec{x} + C) \cap (\vec{y} + C) = \emptyset$

Remark: A code C partitions F^n in the cosets of the elements because belonging to the same set is an **equivalence relation**.

We can now go back to the beginning of the section and look at Bob's strategy for correcting the error:

1. Bob receives $\vec{r}$
2. Bob computes $\vec{s}(\vec{r})$
3. If $\vec{s} = 0$ then there is no error and we are done
4. If $\vec{s} \neq 0$ then Bob has to look at $\vec{r} + C$ and find the element with the minimum weight.

Example: Given the $[4, 7]$ binary Hamming Code

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{13.28}$$

Syndrome	Coset leader
$(0,0,0)^\mathsf{T}$	$(0,0,0,0,0,0,0)$
$(0,0,1)^\mathsf{T}$	$(0,0,0,0,0,0,1)$
$(0,1,0)^\mathsf{T}$	$(0,0,0,0,0,1,0)$
$(1,0,0)^\mathsf{T}$	$(0,0,0,0,1,0,0)$

Syndrome	Coset leader
$(0,1,1)^\mathsf{T}$	$(0,0,0,1,0,0,0)$
$(1,0,1)^\mathsf{T}$	$(0,0,1,0,0,0,0)$
$(1,1,0)^\mathsf{T}$	$(1,0,0,0,0,0,0)$
$(1,1,1)^\mathsf{T}$	$(0,1,0,0,0,0,0)$

From the second row we get:

$$H\vec{e} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \Longleftrightarrow \begin{cases} e_1 + e_2 + e_3 + e_5 = 0 \\ e_1 + e_2 + e_4 + e_6 = 0 \\ e_2 + e_3 + e_4 + e_7 = 1 \end{cases} \tag{13.29}$$

The system of equations above is obtained by picking e_i from the elements of H and must be solved finding the solution with the minimal number of "1"s, i.e. the coset leader which, in this case is $(0,0,0,0,0,0,1)$, in accordance with the table above.

Definition 13.14 (Coset leader): A word of minimal weight in a coset is called a *coset leader*. (N.B.: there could be more than a coset leader for a given coset)

Exercise 13.3: Let $C \subseteq \mathbb{Z}_2^6$ be the $[6, 3]$ linear code with generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{13.30}$$

a) List the elements of C
b) Find the minimum distance of C
c) How many errors will C correct?
d) Find a check matrix for C
e) Make a syndrome chart for C
f) Use the syndrome chart to correct the following errors:
   i) (110101)
   ii) (010111)
   iii) (110111)

Solution:

a) Every element of C is a linear combination of the elements of the basis (the rows): $x \in C \iff \exists \alpha_1, \alpha_2, \alpha_3$ s.t. $\vec{x} = \vec{\alpha}G = \alpha_1\vec{g_1} + \alpha_2\vec{g_2} + \alpha_3\vec{g_3}$ where $\vec{g_i}$ are the rows of G.
   So there are $8 = 2^3$ elements where 2 are the choices between 0 and 1 and 3 are the (linearly independent) lines.

$\vec{\alpha}$	$\vec{x}$
(000)	(000000)
(001)	(001111)

$\vec{\alpha}$	$\vec{x}$
(010)	(010011)
(011)	(011100)

$\vec{\alpha}$	$\vec{x}$
(100)	(100110)
(101)	(101001)

$\vec{\alpha}$	$\vec{x}$
(110)	(110101)
(111)	(111010)

   we can notice that all the elements are different since all the rows of G are linearly independent.
b) Since the code is linear, we can consider just the distance from 0 (i.e. the minimal number of "1"s). This gives $d_{\min} = 3$
c) $\left\lfloor \frac{3-1}{2} \right\rfloor = 1$
d) The check matrix is the generator of $C^\perp$ so we need to find a basis of $C^\perp$:

$$
\begin{aligned}
&\vec{x} \in C^{\perp} \\
\iff & x \perp \vec{g_j} \forall j = 1,2,3 \\
\iff & G\vec{x}^{\mathsf{T}} = 0 \\
\iff & \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\
\iff & \begin{cases} x_1 + x_4 + x_5 = 0 \\ x_2 + x_5 + x_6 = 0 \\ x_3 + x_4 + x_5 + x_6 = 0 \end{cases} \rightarrow \begin{cases} x_1 = x_4 + x_5 \\ x_2 = x_5 + x_6 \\ x_3 = x_4 + x_5 + x_6 \end{cases}
\end{aligned}
\tag{13.31}
$$

x_4, x_5 and x_6 can be chosen freely (but they must be linearly independent) and, starting from them, we can calculate the other variables:

$$
\begin{pmatrix} x_4 & x_5 & x_6 \\ (1 & 0 & 0) \\ (0 & 1 & 0) \\ (0 & 0 & 1) \end{pmatrix} \implies \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ (1 & 0 & 1 & 1 & 0 & 0) \\ (1 & 1 & 1 & 0 & 1 & 0) \\ (0 & 1 & 1 & 0 & 0 & 1) \end{pmatrix} = H \tag{13.32}
$$

We can notice that, similarly to Equation (13.14) [p. 65] we end up with an identity matrix inside H.

e) To draw the syndrome chart, we have to find the coset leader(s) associated to each syndrome. This means solving the system $H\vec{e}^{\mathsf{T}} = s_i$ for all possible syndromes s_i and picking the solution(s) with the least amount of "1"s.

As an example we can find the coset leader for $s_2 = (001)^{\mathsf{T}}$:

$$
\begin{aligned}
& H\vec{e}^{\mathsf{T}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
& \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} e_1 \\ \vdots \\ e_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
& \begin{cases} e_1 + e_3 + e_4 = 0 \\ e_1 + e_2 + e_3 + e_5 = 0 \implies e_6 = 1 \\ e_2 + e_3 + e_6 = 1 \end{cases}
\end{aligned}
\tag{13.33}
$$

Which gives (000001) as a coset leader. The same process can be repeated to obtain all coset leaders:

$\vec{s}^{\mathsf{T}}$	C.L.
(000)	(000000)
(001)	(000001)
(010)	(000010)

$\vec{s}^{\mathsf{T}}$	C.L.
(011)	(010000)
(100)	(000100)

$\vec{s}^{\mathsf{T}}$	C.L.
(101)	(000101)
	(110000)
	(001010)

$\vec{s}^{\mathsf{T}}$	C.L.
(110)	(100000)
(111)	(001000)

f) To correct the errors we simply have to find the syndrome associated to the message Bob received and then flip the bit (if any) corresponding to the "1" in the coset leader associated with the syndrome.

If Bob receives $\vec{x}_1 = (110101)$:

$$\vec{s}(\vec{x}_1) = H \cdot \vec{x}_1^{\mathsf{T}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{13.34}$$

Since the coset leader for $(0,0,0)^{\mathsf{T}}$ is $(0,0,0,0,0,0)$ this means that there was no error.

If Bob receives $\vec{x}_2 = (010111)$:

$$\vec{s}(\vec{x}_2) = H \cdot \vec{x}_2^{\mathsf{T}} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{13.35}$$

In this case the coset leader is $\vec{e_2} = (0,0,0,1,0,0)$ which gives us the correct message:

$$\vec{c}_2 = \vec{e}_2 + \vec{x}_2 = (0,1,0,0,1,1) \tag{13.36}$$

14. BB84

14.1. Steps

1. Alice generates 2 random sequences:

$$\begin{aligned} A &= (a_1, ..., a_n) \quad a_j \in \{0, 1\} \\ S &= (s_0, ..., s_n) \quad s \in \{+, \times\} \end{aligned} \tag{14.1}$$

2. Alice sends photons to Bob in this way:
   - If $s_j = +$ Alice filters the j^{th} photon with the polarizer oriented along $+$
   - If $a_j = 0$ and the j^{th} photon exits through the ordinary ray, Alice keeps the photon
   - If $a_j = 0$ and the j^{th} photon exits through the extraordinary ray, Alice destroys the photon and tries again
   - Analogously for $a_j = 1$ and/or $s_j = \times$
3. Before receiving the photons, Bob generates a random sequence $R = (r^1, ..., r^n)$ with $r^j \in \{+, \times\}$. He then filters the incoming photons using a polarizer oriented according to r^j and records the digit 0 if the photon exits through the ordinary ray and 1 otherwise. In this way he obtains the sequence

$$B = (b_1, ..., b_n) \quad b_j \in \{0, 1\} \tag{14.2}$$

4. Alice and Bob publish the sequences S and R and throw away the bits of A and B corresponding to different choices of the basis, obtaining A' and B'. On average, the length of A' and B' is $n/2$. In an ideal world, at this point, Alice and Bob share a secret key without never having to meet.
5. There are 2 sources of non-ideality:
   i. Photons can be lost or perturbed during transmission
   ii. Someone (Eve) can intercept the communication
6. The previous non-idealities imply that some of the bits in B' present a small fraction of errors. Depending on the amount of errors, Alice and Bob can detect the presence of Eve.

14.2. One possible attack

Eve could intercept the photons before they arrive to Bob, measure them and them send to Bob a new photon. The problem for Eve is that she doesn't know what basis Alice used, so she must chose at random. If Eve performs this attack, the probability that $A' \neq B'$ is $n_e/4$, where n_e is the number of photons that were eavesdropped.

To detect this kind of attacks, Alice and Bob can sacrifice a small portion of the their qubits, count the discrepancies, measure the number of errors and the proceed. For the following, it is important to estimate the amount of information that Eve can obtain.

Definition 14.1 (Rényi entropy): We want a function that embodies our ignorance: if we have a binary process with 2 possible outcomes, characterized by the probabilities p_0 and $p_1 = 1 - p_0$, when $p_0 = 1/2$, the ignorance is maximal.

$$H_R = -\log_2(p_0^2 + p_1^2) \tag{14.3}$$

In a more general form:

$$H_R = -\log_2\left(\sum_{\vec{a}} p^2(\vec{a})\right) \tag{14.4}$$

Definition 14.2 (Rényi information): Let $\vec{a}$ be a string of length n, then the Rényi information is defined as:

$$I_R = n - H_R \tag{14.5}$$

Definition 14.3 (Shannon entropy):

$$H_S(x) \equiv -\sum_x p_x \log_2 p_x \tag{14.6}$$

14.3. Example: Loepp-Wooters

Suppose that Eve intercepts a photon using the basis $\{|m_1\rangle, |m_2\rangle\} = \left\{\begin{pmatrix}1/2\\ \sqrt{3}/2\end{pmatrix}, \begin{pmatrix}\sqrt{3}/2\\ -1/2\end{pmatrix}\right\}$ and obtains the state $|m_1\rangle$.

When Alice publishes S, Eve discovers that Alice used the basis M_+ to filter the photons. How much information did Eve gain on the original state of the photon?

We must compute the probability that the state of the photon released by Alice had horizontal polarization, knowing that, when filtered by Eve, its polarization was turned to $|m_1\rangle$.

Notice that this is not the quantum transition probability (QPT). In fact, QPT is

$$\mathcal{P}(|m_1\rangle \mid |\leftrightarrow\rangle) = \text{probability of having } |m_1\rangle \tag{14.7}$$

after the measurement, given that, before the measurement, it was $|\leftrightarrow\rangle$. We find out that $\mathcal{P}(|m_1\rangle \mid |\leftrightarrow\rangle) \equiv |\langle m_1|\leftrightarrow\rangle|^2$.

What we want now is the opposite, namely $\mathcal{P}(|\leftrightarrow\rangle||m_1\rangle)$. We can compute it using Bayes, formula:

$$\mathcal{P}(|\leftrightarrow\rangle||m_1\rangle) = \underbrace{\mathcal{P}(|m_1|\leftrightarrow\rangle)}_{1/4}\frac{\overbrace{\mathcal{P}(\leftrightarrow)}^{1/4}}{\mathcal{P}(m_1)} \tag{14.8}$$

The remaining probability $\mathcal{P}(m_1)$ is the probability that Eve obtains $|m_1\rangle$ after measurement,computed as if we were Eve, who doesn't know the state of the photon. According to her, the photon is in a mixed state of $\leftrightarrow$ and $\updownarrow$:

$$\rho_{\text{eve}} = \frac{1}{2}|\leftrightarrow\rangle\langle\leftrightarrow| + \frac{1}{2}|\updownarrow\rangle\langle\updownarrow| \tag{14.9}$$

The probability of finding $|m_1\rangle$ is thus:

$$\text{Tr}\Big(\rho_{\text{eve}} \cdot \mathcal{P}_{m_1}\Big) = \left(\frac{1}{2}|\leftrightarrow\rangle\langle\leftrightarrow| + \frac{1}{2}|\updownarrow\rangle\langle\updownarrow|\right)(|m_1\rangle\langle m_1|) = \ldots = \frac{1}{2} \tag{14.10}$$

So, $p_0 = \mathcal{P}(\leftrightarrow \mid m_1) = \frac{1}{4}$ and $p_1 = \mathcal{P}(\updownarrow \mid m_1) = \frac{3}{4}$. This gives a Rényi entropy:

$$H_R = -\log\left(\frac{1}{16} + \frac{9}{16}\right) = 0.678 \tag{14.11}$$

Since, without Eve's intervention, we had $H_R = 1$, we can calculate the gain in information, which is equal to the loss in entropy as: $H_{R\text{ initial}} - H_{R\text{ final}} = 0.322$ bits.

Theorem 14.1 (No cloning):

1. $\nexists$ a unitary operator on $H \otimes H$ s.t.

$$U(|\psi\rangle|0\rangle) = |\psi\rangle|\psi\rangle e^{i\varphi(\psi)} \tag{14.12}$$

2. There exists a unitary operator U on $H \otimes H$ such that, given a specific set $\{|\psi_j\rangle, j \in \mathbb{N}\}$,

$$U(|\psi_j\rangle|0\rangle) = |\psi_j\rangle|\psi_j\rangle \tag{14.13}$$

if and only if $\{|\psi_j\rangle, j \in \mathbb{N}\}$ form an orthonormal set.

The no cloning theorem prevents Eve from being able to duplicate information sent by Alice.

14.4. Teleportation strategy

Alice and Bob share an entangled pair:

1. $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
2. Alice owns a further two level-system in the (possibly unknown) state $|s\rangle = \alpha|0\rangle + \beta|1\rangle$. Alice aims at conveying $|s\rangle$ to Bob. The three particle state is

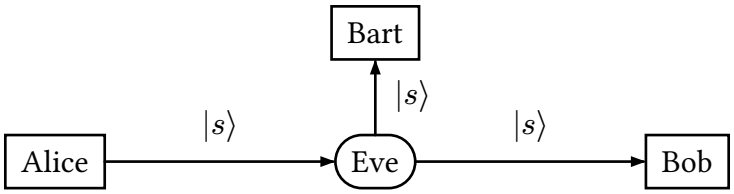
$$\begin{aligned} |s\rangle \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{2}\Bigg[& |\Phi^+\rangle \otimes (\alpha|0\rangle + \beta|1\rangle) + \\ & + |\Phi^-\rangle \otimes (\alpha|0\rangle - \beta|1\rangle) + \\ & + |\Psi^+\rangle \otimes (\beta|0\rangle + \alpha|1\rangle) + \\ & + \underbrace{|\Psi^-\rangle}_{A} \otimes \underbrace{(-\beta|0\rangle + \alpha|1\rangle)}_{B} \Bigg] \end{aligned} \tag{14.14}$$

The set $\{|\Phi^+\rangle, |\Phi^-\rangle, |\Psi^+\rangle, |\Psi^-\rangle\}$ is called a *Bell's basis* of the two particle space. Notice that such states are attributed to Alice's pair.

3. Alice measures a non-degenerate observable that has Bell's basis as a set of eigenvectors
4. Alice tells Bob her result. N.B.: this communication is not made through the entangled pair, it is "classical" and not superluminal
5. Depending on Alice's result, Bob applies the following gates on his qubit:

$$\begin{aligned} |\Phi^+\rangle &\to \mathbb{I} \\ |\Phi^-\rangle &\to \sigma_z \\ |\Psi^+\rangle &\to \sigma_x \\ |\Psi^-\rangle &\to \sigma_z\sigma_x \end{aligned} \tag{14.15}$$

Remark: At the end the state of the qubit initially in $|s\rangle$ is an entangled state with the other qubit of Alice. So, individually it is a mixed state. Sending it to Bob provokes perturbation exactly like measuring it.
The teleportation strategy gives no advantage to Eve. In fact the cloning strategy was:



But this cannot work because after teleportation, Eve has not $|s\rangle$ to send to Bob. The idea was "cloning at a distance + ancilla" (the ancilla is the second qubit of Alice). Even for this setting a no-cloning result can be proved.

14.5. Obtaining a secret shared key

As stated before, after Alice and Bob have obtained the sequences A and B, they are still far from having a secret shared keys due to the following problems that cause $A \neq B$:

1. Alice and Bob chose different basis and thus have a probability $P = 1/4$ of ending up with the same bit in a given position
2. The channel is not perfect and can thus introduce errors
3. Eve can perturb the channel and introduce additional errors

The first issue is straightforward to solve since all is required to do is compare publically the sequence of the basis and discard the bits corresponding to different base choices. This results in a reduction of the total number of bits:

$$\begin{aligned} A &\longrightarrow A' = (a'_1, ..., a'_{n'}) \\ B &\longrightarrow B' = (b'_1, ..., b'_{n'}) \\ n &\approx \frac{n}{2} \end{aligned} \tag{14.16}$$

It is important to notice that, at this point, A', B' are **neither identical nor private** due to issues 2. and 3.

Starting from A' and B' we need to use **error correction** and **privacy amplification** to constructs two identical and secure sequences. Steps 5 and 6 of BB84 need to be modified to account for these changes:

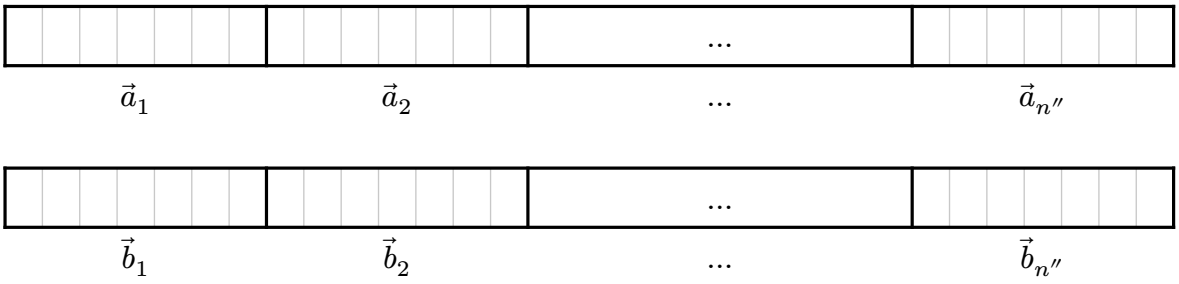
a) Check some of the bits of A' and B' to estimate the probability of error and then discard these bits
b) Correct the errors in the remaining bits, obtaining two sequences that are (hopefully) identical but not private
c) Perform **privacy amplification** to extract a secure substring starting from an insecure one

14.5.1. Estimating the probability of error

If we suppose that B' is composed of 10000 bits, then Bob can select 100 of them and compare them publically with the corresponding bits of A'. Supposing that there are 2 errors in the 100-bits sequences, Alice and Bob can estimate $P(\text{error}) = 2\%$ (without accounting for Eve's contribution). Obviously, all 100 bits used for the comparison have to be discarded after this step, giving A'' and B'' of length $n'' = 9900$ bits.

14.5.2. Error correction (using Hamming)

Continuing with the example above (e.g. $P(\text{error}) = 2\%$), if we consider strings which are 50 bits long, we will have, on average, 1 error per string. Since we want to work with strings which have a probability of error $\ll 1$, we can consider strings of length 7 and split A'' and B'' in 7 bits long strings:



We can now take the check matrix for the $[7, 4]$ binary Hamming code and compute the syndromes:

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{14.17}$$

Alice can compute $\vec{s}^A = H\vec{a}^\mathsf{T}$ and send it to Bob. Bob, on the other hand, can compute $\vec{s}^B = H\vec{b}^\mathsf{T}$ and compare it with $\vec{s}^A$. I we define $\vec{e} = \vec{b} - \vec{a}$ we get

$$H\vec{e}^\mathsf{T} = H\vec{b}^\mathsf{T} - H\vec{a}^\mathsf{T} = \underbrace{\vec{s}^B - \vec{s}^A}_{\text{known by Bob}} \tag{14.18}$$

At this point, Bob can get $\vec{s}^E = H\vec{e}^\mathsf{T}$; by computing the syndrome chart for the $[7, 4]$ Hamming code, we can see that every coset has exactly one coset leader and this means that the error can be uniquely determined (provided that we are sure that we have at most one error in $\vec{b}$).

It is important to notice that, during this procedure, Eve can gain the knowledge of both H and $\vec{s}^A$, which is to say:

$$\begin{aligned} a_1 + a_2 + a_3 + a_5 &= \vec{s}_1^A \\ a_1 + a_2 + a_4 + a_6 &= \vec{s}_2^A \\ a_2 + a_3 + a_4 + a_7 &= \vec{s}_3^A \end{aligned} \tag{14.19}$$

Example:

$$\begin{aligned} \vec{a} &= (1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1) \\ \vec{b} &= (1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1) \end{aligned} \tag{14.20}$$

1. Alice computes $\vec{s}^A = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\mathsf{T}$ and sends it to Bob
2. Bob computes $\vec{s}^B = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^\mathsf{T}$
3. Bob computes

$$\vec{s}^E = \vec{s}^B - \vec{s}^A = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = H\vec{e}^\mathsf{T} \tag{14.21}$$

   which means that $\vec{e}$ is the coset leader of $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\mathsf{T}$ which is $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
4. Bob can, finally, flip the second bit and correct the error.

The advantage of this strategy is that Alice sends only one information to Bob for each sub-block and Bob doesn't send anything back.

The drawback is that computing the syndrome involves matrix product, which can be costly for large matrices.

14.5.3. Error correction (using parity check)

Another strategy that Alice and Bob can adopt is the following:

1. Alice and Bob must agree on a permutation of bits which has to be applied to the whole string to randomize the distribution of errors (this is useful in cases where errors are concentrated in a small section of the string).
2. Similarly to before, Alice and Bob divide the string in blocks of a suited size so that the probability of error in each block is $\ll 1$.
3. They compute the parity of each block and publically compare the outcomes. I the parity is the same, Bob leaves the block untouched, otherwise the affected block gets divided into equal parts and the procedure starts again, similarly to binary search. This process will get to an end when they find the affected bit.
4. When the error is found, Bob flips the corresponding bit.
5. The process continues until all the string has been error-corrected, with the difference that, the more we proceed, the fewer the errors and thus larger blocks can be used.

Using this strategy, Eve gets to know the parity of all sub-blocks which, like in the previous case, are a linear combination of Alice's bits.

14.5.4. Privacy amplification

At this point in the protocol, the strings shared by Alice and Bob are identical but we want them also to be private, which is to say that, to Eve, all strings should be equiprobable.

Example: Alice and Bob know that Eve knows exactly one bit of $\vec{a} = (a_1, a_2, a_3)$ but they don't know which one it is.

We can define

$$\vec{\alpha} \triangleq (a_1 + a_2, a_2 + a_3) \tag{14.22}$$

It is easy to see (looking at the "truth table"), that Eve doesn't know anything about $\vec{\alpha}$, which is to say that all αs are equiprobable to her.

α is called a **privacy amplification scheme** because Eve went from knowing something about the information shared between Alice and Bob to knowing nothing.

Such a scheme can be represented by the matrix

$$\begin{gathered} G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \\ \vec{\alpha}^{\mathsf{T}} = G\vec{a}^{\mathsf{T}} \end{gathered} \tag{14.23}$$

Example: Similarly to before, we know that Eve knows **two** bits of $\vec{a} = (a_1, a_2, a_3, a_4)$ but we don't know which ones. We decide to try with the scheme

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \tag{14.24}$$

which gives the following:

$$\vec{\alpha} = (a_1 + a_2 + a_3, a_2 + a_3 + a_4) \tag{14.25}$$

If we suppose that the bits known by Eve are a_1 and a_4 it is easy to see that Eve can compute $a_1 + a_2 = a_1 + \cancel{a_2} + \cancel{a_3} + \cancel{a_2} + a_4 + \cancel{a_3}$.

Differently from before, Eve now **knows** a linear combination of a_1 and a_2, this means that, in this case, for Eve, not all αs are equiprobable and thus **this amplification scheme fails**.

If we look at the minimal weight of the code generated by G, we see that it is 2, given by $\begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}$ and 2 is also equal to the information (in the sense of the number of bits) known by Eve.

Theorem 14.2: Suppose $\vec{a} \in \mathbb{Z}_2^n$ and that Eve knows exactly t bits of $\vec{a}$. Let G generate a linear $[n, k]$ code with minimum weight w.

Let $\vec{\alpha}^\mathsf{T} = G\vec{a}^\mathsf{T}$ a privacy amplification scheme, then:

1. If $w > t$, then Eve knows **nothing** about $\vec{\alpha}$
2. If $w \leq t$, then Eve could know **something** about $\vec{\alpha}$ (in the sense that there exists a sequence of t bits that would allow Eve to know something about $\vec{\alpha}$)

Let's now consider the more realistic case in which Eve knows some linear combination of the bits a_i as Eve typically knows the syndrome of Alice $\vec{s}^A$.

Example: Linear binary $[7, 4]$ code:

$$\begin{aligned}\vec{s}^A = H\vec{a}^\mathsf{T} &= \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \vec{a}^\mathsf{T} \\ &= \begin{pmatrix} a_1 + a_2 + a_3 + a_5 \\ a_1 + a_2 + a_4 + a_6 \\ a_2 + a_3 + a_5 + a_7 \end{pmatrix} \\ &= \begin{pmatrix} s_1^A \\ s_2^A \\ s_3^A \end{pmatrix}\end{aligned} \tag{14.26}$$

Eve knows three linear combinations:

We do some trials to see what happens and how many bits we need to sacrifice.

1. Drop a_4, a_6, a_7 and remain with $\alpha = (a_1, a_2, a_3, a_5)$. This is not good because s_1^A is left untouched and known by Eve $\implies$ **all the lines of $\vec{s}^A$** must be involved.

$$\begin{pmatrix} a_1 + a_2 + a_3 + a_5 \\ a_1 + a_2 + \cancel{a_4} + \cancel{a_6} \\ a_2 + a_3 + a_5 + \cancel{a_7} \end{pmatrix} \leftarrow \text{Untouched} \tag{14.27}$$

2. Drop a_1, a_2, a_7, now we affect all the rows but the first two are affected in the same way and thus Eve can gain information by summing $s_1^A + s_2^A \implies$ **we must "touch" all equations and all linear combinations of them**.

$$\text{Affected in the same way} \rightarrow \begin{pmatrix} \cancel{a_1} + \cancel{a_2} + a_3 + a_5 \\ \cancel{a_1} + \cancel{a_2} + a_4 + a_6 \\ \cancel{a_2} + a_3 + a_5 + \cancel{a_7} \end{pmatrix} \tag{14.28}$$

3. Drop a_5, a_6, a_7. Now it works.

$$\begin{pmatrix} a_1 + a_2 + a_3 + \cancel{a_5} \\ a_1 + a_2 + a_4 + \cancel{a_6} \\ a_2 + a_3 + \cancel{a_5} + \cancel{a_7} \end{pmatrix} \tag{14.29}$$

Theorem 14.3: Suppose that Eve has $I_R < t$ and $t < n$, where $I_R \approx$ "number of bits known" and n is the number of bits of $\vec{a}$.

Let $s < n - t$ and $k = n - t - s$.

Let $\vec{\alpha}^\mathsf{T} = G\vec{a}^\mathsf{T}$ where G is a randomly chosen $k \times n$ matrix. Then Eve's average Rényi information (Definition 14.2 [p. 73]) about α is

$$I_R^E \leq \frac{2^{-s}}{\log(2)} \tag{14.30}$$

Which tells us that the more bits we sacrifice, the less Eve knows.

$$\vec{a} = \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad}^{n}$$
$$\text{Final string} = \qquad\qquad\qquad\qquad\qquad\qquad \text{s}$$
$$\text{t} \tag{14.31}$$

Moreover, Shannon's entropy $\geq H_R$

We will now look at some examples that show us the relationship between the information available to Eve and the Shannon's and Rényi's entropies.

Example (No information): If Eve has access to no information at all, then every $\vec{a}$ is equiprobable. Thus

$$\begin{aligned} H_R &= \log_2 \sum_{\vec{a}} \left(\frac{1}{2^n}\right)^2 \\ &= -\log_2 \sum_{\vec{a}} \frac{1}{2^{2n}} \\ &= -\log_2 \frac{2^n}{2^{2n}} \\ &= n \implies I_R = n - n = 0 \end{aligned} \tag{14.32}$$

Example (One string is known): In this case there is one $\vec{a}^*$ with probability 1 and all the other have probability 0:

$$\begin{aligned} H_R = 0 &\implies I_R = n \\ H_S = 0 &\implies I_S = n \end{aligned} \tag{14.33}$$

Example (Intermediate case): We have one $\vec{a}^*$ with probability $1/2$ and all the other strings have probability

$$p(\vec{a}) = \frac{1/2}{2^n - 1} = \frac{1}{2^{n+1} - 2} \tag{14.34}$$

In this case we get the following:

$$
\begin{aligned}
H_R &= -\log_2\left(p^2(\vec{a}^*) + \sum_{\vec{a}\neq\vec{a}^*} p^2(\vec{a})\right) \\
&= -\log_2\left(\frac{1}{4} + \frac{1}{\left(2^{n+1}-2\right)^2}(2^n-1)\right) \\
&= -\log_2\left(\frac{1}{4} + \frac{1}{4}\cdot\frac{1}{2^n-1}\right) \\
&= -\log_2\left[\frac{1}{4}\left(1+\frac{1}{2^n-1}\right)\right] \\
&= -\log_2 2^{n-2} - \log_2\frac{1}{2^n-1} \\
&= 2 - n + \log_2(2^n-1)
\end{aligned}
\tag{14.35}
$$

We can compute H_R for different values of n to see what we get:

- For $n=3$: $H_R = 1.8 \implies I_R = 1.2$
- For $n=9$: $H_R = 1.997 \implies I_R = ???$
- For $n\to\infty$:

$$
\begin{aligned}
H_R &= 2 + \overbrace{\log_2\frac{1}{2^n}}^{n} + \log_2(2^n-1) \\
&= 2 + \underbrace{\log_2\frac{2^n-1}{2^n}}_{\to 0 \text{ as } n\to\infty} \to 2 \\
I_R &\to \infty
\end{aligned}
\tag{14.36}
$$

15. Quantum error correction

15.1. Introduction to Shor's correcting code

15.1.1. X-correcting code

Problem: protecting the state of one qubit $|s\rangle = a|0\rangle + b|1\rangle, a, b \in \mathbb{C}, |a|^2 + |b|^2 = 1$

Meaning that if the state is corrupted (modified by accident in an uncontrollable way), then we are able to recover it again, i.e. recover a and b. Every possible accident is represented by a 2×2 unitary matrix.

We can:
1. Add other qubits
2. Apply any unitary transformation on the whole system
3. Perform measurements: projective and "incomplete" (i.e. the eigenspaces are not one-dimensional)

15.1.2. 1st step: X-correcting code

We suppose that the only possible error is described by:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \begin{cases} \sigma_x|0\rangle = |1\rangle \\ \sigma_x|1\rangle = |0\rangle \end{cases} \tag{15.1}$$

Preliminary step: $|s\rangle \to |s\rangle \underbrace{|00\rangle}_{\substack{\text{Two new} \\ \text{qubits}}} = a|000\rangle + b|100\rangle$

Now we use a unitary transformation that maps

$$|000\rangle \to |000\rangle \qquad |100\rangle \to |111\rangle \tag{15.2}$$

(the action on the other elements of the computational basis is irrelevant) so:

$$|s_c\rangle = a|000\rangle + b|111\rangle \quad \text{quantum coding} \tag{15.3}$$

The error σ_x affects one of the three qubits but we don't know which one.

$$|s_c\rangle = \begin{cases} \sigma_x \text{ on 1st} \to a|100\rangle + b|011\rangle = |s_{c1}\rangle \\ \sigma_x \text{ on 2nd} \to a|010\rangle + b|101\rangle = |s_{c2}\rangle \\ \sigma_x \text{ on 3rd} \to a|001\rangle + b|110\rangle = |s_{c3}\rangle \end{cases} \tag{15.4}$$

We measure the observable O whose eigenspaces are:

$$\begin{aligned} &\lambda = 1, \quad E_1 = \text{span}\{|100\rangle, |011\rangle\} \Rightarrow P_1 = |100\rangle\langle 100| + |011\rangle\langle 011| \\ &\lambda = 2, \quad E_2 = \text{span}\{|010\rangle, |101\rangle\} \Rightarrow P_2 = |010\rangle\langle 010| + |101\rangle\langle 101| \\ &\lambda = 3, \quad E_3 = \text{span}\{|001\rangle, |110\rangle\} \Rightarrow P_3 = |001\rangle\langle 001| + |110\rangle\langle 110| \\ &\lambda = 4, \quad E_4 = \text{span}\{|000\rangle, |111\rangle\} \Rightarrow P_4 = |000\rangle\langle 000| + |111\rangle\langle 111| \end{aligned} \tag{15.5}$$

Suppose now that σ_x affected the 1st qubit, after the measurement, we obtain $\lambda = 1$ so we discover that the wrong bit is the first.

We apply σ_x on the first qubit and obtain

$$|s_c\rangle = (\sigma_x \otimes \mathbb{I} \otimes \mathbb{I})|s_{c1}\rangle \tag{15.6}$$

The same procedure holds in the case the error affects the other qubits or in the case there is no error to correct.

Remark:
1. We always assume that only 1 qubit is affected by the error
2. The measurement did not perturb $|s_c\rangle$ but only revealed which bit was affected
3. This protocol can correct more general errors

Consider an error represented by:

$$U = \begin{pmatrix} \cos(\theta) & i\sin(\theta) \\ i\sin(\theta) & \cos(\theta) \end{pmatrix} \qquad \theta \in [0, 2\pi) \tag{15.7}$$

Assume e.g. U affects the 2nd qubit:

As before

$$\begin{aligned} &|s\rangle \to |s_C\rangle \to (\mathbb{I} \otimes U \otimes \mathbb{I})|s_C\rangle = \\ &= a(|0\rangle U|0\rangle|0\rangle) + b(|1\rangle U|1\rangle|1\rangle) \\ &= a(|0\rangle(\cos(\theta)|0\rangle + i\sin(\theta)|1\rangle)|0\rangle) + b(|1\rangle(\cos(\theta)|1\rangle + i\sin(\theta)|0\rangle)|1\rangle) \\ &= a(\cos(\theta)|000\rangle + i\sin(\theta)|010\rangle) + b((\cos(\theta)|111\rangle + i\sin(\theta)|101\rangle) \end{aligned} \tag{15.8}$$

Applying the same observable as before, two outcomes are possible: $\lambda = 2$ and $\lambda = 4$

If $\lambda = 2$:

$$\begin{aligned} \frac{P_2|s'_C\rangle}{\|P_2\|s'_C\rangle} &= \frac{i\sin(\theta)(a|010\rangle + b|101\rangle)}{\sqrt{\sin^2(\theta)(|a|^2 + |b|^2)}} = \\ &= i\frac{\sin(\theta)}{|\sin(\theta)|}(a|010\rangle + b|101\rangle) \sim a|010\rangle + b|101\rangle \end{aligned} \tag{15.9}$$

Applying σ_x to the 2nd qubit, we get $|s_C\rangle$

If $\lambda = 4$:

$$\frac{P_4|s'_C\rangle}{\|P_4\|s'_C\rangle} = \frac{\cancel{\cos(\theta)}(a|0\rangle + b|1\rangle)}{\cancel{|\cos(\theta)|}} \cong a|000\rangle + b|111\rangle = |s_C\rangle \tag{15.10}$$

Remark: The protocol cannot correct Z-type errors:

$$\mathbb{I} \otimes \mathbb{I} \otimes \sigma_z|s_C\rangle = a|000\rangle - b|111\rangle \tag{15.11}$$

The information of which qubit was affected by the error is lost.

Notice that in this case the measurement modified the state

15.1.3. Quantum correcting code for Z-type errors

$$
\begin{gathered}
|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
\sigma_z|+\rangle = |-\rangle \quad \sigma_z|-\rangle = |+\rangle
\end{gathered}
\tag{15.12}
$$

We start from $|s\rangle = a|0\rangle + b|1\rangle$ with a unitary transformation

$$
\begin{aligned}
H : |0\rangle &\to |+\rangle \\
|1\rangle &\to |-\rangle
\end{aligned}
\tag{15.13}
$$

So the state becomes: $|s'\rangle = a|+\rangle + b|-\rangle$

Adding two qubits and proceeding like in the previous case $|s_C\rangle = a|{+}{+}{+}\rangle + b|{-}{-}{-}\rangle$

Considering an another observable:

$$
\begin{aligned}
&\mu = 1, \quad Q_1 = |{-}{+}{+}\rangle\langle{-}{+}{+}| + |{+}{-}{-}\rangle\langle{+}{-}{-}| \\
&\mu = 2, \quad Q_2 = |{+}{-}{+}\rangle\langle{+}{-}{+}| + |{-}{+}{-}\rangle\langle{-}{+}{-}| \\
&\mu = 3, \quad Q_3 = |{+}{+}{-}\rangle\langle{+}{+}{-}| + |{-}{-}{+}\rangle\langle{-}{-}{+}| \\
&\mu = 4, \quad Q_4 = |{+}{+}{+}\rangle\langle{+}{+}{+}| + |{-}{-}{-}\rangle\langle{-}{-}{-}|
\end{aligned}
\tag{15.14}
$$

Example: This protocol corrects one error of the kind

$$
V = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}
\tag{15.15}
$$

In the basis $\{|0\rangle, |1\rangle\}$

Writing V in the basis $\{|+\rangle, |-\rangle\}$

$$
\begin{aligned}
V|+\rangle = V\frac{|0\rangle + |1\rangle}{\sqrt{2}} &= \frac{|0\rangle + e^{i\phi}|1\rangle}{\sqrt{2}} \\
&= \frac{|+\rangle + |-\rangle + e^{i\phi}|+\rangle - e^{i\phi}|-\rangle}{2} = \\
&= \frac{1 + e^{i\phi}}{2}|+\rangle + \frac{1 - e^{i\phi}}{2}|-\rangle \\
&\sim \cos\left(\frac{\phi}{2}\right)|+\rangle - i\sin\left(\frac{\phi}{2}\right)|-\rangle
\end{aligned}
\tag{15.16}
$$

$$
\begin{aligned}
V|-\rangle &= \frac{1 - e^{i\phi}}{2}|+\rangle + \frac{1 + e^{i\phi}}{2}|-\rangle \\
&\sim \cos\left(\frac{\phi}{2}\right)|-\rangle - i\sin\left(\frac{\phi}{2}\right)|+\rangle
\end{aligned}
\tag{15.17}
$$

So the the matrix can be written as:

$$V' = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & -i\sin\left(\frac{\phi}{2}\right) \\ -i\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{pmatrix} \tag{15.18}$$

which is the same example made for the X-correcting code (see Equation (15.7) [p. 84])

15.2. The Shor code

Now we put the two schemes together to create an error-correction protocol that protects against *all* single-qubit errors[2]. We begin by appending eight additional qubits in a standard state and then perform a unitary transformation so that the state $|0\rangle$ ends up encoded as:

$$|0_C\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \tag{15.19}$$

And the state $|1\rangle$ is encoded as:

$$|1_C\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \tag{15.20}$$

Thus the initial state $|s\rangle = a|0\rangle + b|1\rangle$ is encoded as $|s_C\rangle = a|0_C\rangle + b|1_C\rangle$

Let's see this encoding step by step:
As a first thing we do a "*partial cloning*":

$$\begin{aligned} |0\rangle &\to |000000000\rangle \\ |1\rangle &\to |111111111\rangle \end{aligned} \tag{15.21}$$

And then we use some unitary operators that, analogously with the X-correcting code:

$$\begin{aligned} (|000\rangle)^{\otimes 3} &\to \left(\frac{|000\rangle + |111\rangle}{\sqrt{2}}\right)^{\otimes 3} \\ (|111\rangle)^{\otimes 3} &\to \left(\frac{|000\rangle - |111\rangle}{\sqrt{2}}\right)^{\otimes 3} \end{aligned} \tag{15.22}$$

Thus this code can correct: $X_1, ..., X_9$; $Z_1, ..., Z_9$; $X_1Z_1, ..., X_9Z_9$ errors that apparently are 27 single qubit errors, but:

$$\begin{aligned} Z_1|0_C\rangle &= \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \\ &= Z_2|0_C\rangle = Z_3|0_C\rangle \end{aligned} \tag{15.23}$$

and the same true is for $Z_1|1_C\rangle = Z_2|1_C\rangle = Z_3|1_C\rangle$.

So the possible errors are 21:

$$\underbrace{(X_1, ..., X_9)}_{9}; \quad \underbrace{(Z_1, Z_2, Z_3)}_{1}, \underbrace{(Z_4, Z_5, Z_6)}_{1}, \underbrace{(Z_7, Z_8, Z_9)}_{1}; \quad \underbrace{(X_1Z_1, ..., X_9Z_9)}_{9} \tag{15.24}$$

[2] A possible implementation of the Shor code can be found here°

We need an observable O whose eigenspaces are the spans of $(\vartheta|0_C\rangle, \vartheta|1_C\rangle)$ where ϑ is any of the allowed errors.

How to construct the observable?

We define the subspaces

$$\begin{aligned}
E_{X_1} &= \mathrm{Span}\{X_1|0_C\rangle, X_1|1_C\rangle\} = \\
&= \mathrm{Span}\Big\{ \frac{1}{2\sqrt{2}}(|100\rangle + |011\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle), \\
&\qquad \frac{1}{2\sqrt{2}}(|100\rangle - |011\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \Big\} \\
&\vdots \\
E_{X_9} &= \mathrm{Span}\{..., ...\} \\
&\vdots \\
E_{Z_8} &= \mathrm{Span}\Big\{ \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle - |111\rangle), \\
&\qquad \frac{1}{2\sqrt{2}}(|100\rangle - |011\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle + |111\rangle) \Big\} \\
&\vdots \\
E_{X_5Z_5} &= \mathrm{Span}\Big\{ \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|010\rangle - |101\rangle) \otimes (|000\rangle + |111\rangle), \\
&\qquad \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|010\rangle + |101\rangle) \otimes (|000\rangle - |111\rangle) \Big\} \\
&\vdots
\end{aligned} \tag{15.25}$$

In this way we construct 22 two-dimensional mutually orthogonal subspaces (the listed ones plus $\mathrm{Span}\{|0_C\rangle, |1_C\rangle\}$). But the whole space has 512 (2^9) dimensions. What is the meaning of the missing 490? They describe spaces with more than one error.

15.2.1. Arbitrary one-qubit errors

Proposition: Every unitary matrix U can be written as a linear combination of $\mathbb{I}, \sigma_x, \sigma_y, \sigma_z$, so by a linear combination of $\mathbb{I}, \sigma_x, \sigma_z, \sigma_x\sigma_z$.

So, $\exists t, u, v, w$ s.t. $W = t\mathbb{I}, u\sigma_x, v\sigma_z, w\sigma_x\sigma_z$(where $t, u, v, w \in \mathbb{C}$)

Take $j \in \{1, ..., 9\}$, $|s'_C\rangle = W_j|s_C\rangle$. Measuring with our previous observable, the state precipitates to one of the 22 subspaces $\Rightarrow$ no perturbation is the collapsed state $\Rightarrow$ we gain $|s_C\rangle$.

If we find an eigenvalue corresponding to a subspace related e.g. to X_5Z_5, then we know that the collapsed state is

$$\begin{aligned}
P_{X_5Z_5}W_5|s_C\rangle &= t|s_C\rangle + u\sigma_{X_5}|s_C\rangle + v\sigma_{Z_5}|s_C\rangle + w\sigma_{X_5}\sigma_{Z_5}|s_C\rangle \\
&= \frac{w\sigma_{X_5}\sigma_{Z_5}|s_C\rangle}{|w|} \sim \sigma_{X_5}\sigma_{X_5}|s_C\rangle
\end{aligned} \tag{15.26}$$

That is reversed by applying $\sigma_{X_5}\sigma_{Z_5}$, obtaining $|s_C\rangle$.

Exercise 15.1: Problem 5 from Loepp-Wootters
Consider the code defined by Equation (15.27). Suppose that the last of the three qubits interacts with a fourth qubit from outside the quantum computer. This fourth qubit starts out in the state $\frac{1}{2}(|0\rangle + |1\rangle)$ and the third and fourth qubits are acted upon by the operation CNOT_{43}. (The subscripts indicate that the fourth qubit is the control and the third qubit is the target.) Show that the standard error-correction process for the X-correcting code also corrects this error. (For this problem you will need to use new versions of the projection operators $P_1, ..., P_4$, since the space that needs to be acted upon is now a four-qubit state space. This is a straightforward extension, in which the P operators leave the fourth qubit unaffected. For example, whereas the original version of P_1 projects onto the subspace spanned by $|100\rangle$ and $|011\rangle$, the new version will project onto the four-dimensional subspace spanned by $|1000\rangle, |1001\rangle, |0110\rangle, |0111\rangle$. In other words, the new version of P_1 is $P_1 \otimes \mathbb{I}$, where $\mathbb{I}$ is the identity operator on the fourth qubit.

Solution:

$$|s_c\rangle = a|000\rangle + b|111\rangle \tag{15.27}$$

$$\begin{aligned} &|s_c\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ &\quad = \left(\frac{1}{\sqrt{2}}\right)(a|0000\rangle + b|1110\rangle + a|0001\rangle + b|1111\rangle) \\ &\quad \Downarrow \text{cnot} = \text{error} \\ &\quad = \frac{a|0000\rangle + b|1110\rangle}{\sqrt{2}} + \frac{a|0011\rangle + b|1101\rangle}{\sqrt{2}} = |s_c'\rangle \end{aligned} \tag{15.28}$$

$$\begin{aligned} P_1 &= \text{error on 1st qubit} = (|100\rangle\langle 100| + |011\rangle\langle 011|) \otimes \mathbb{I}_4 \\ P_2 &= \text{error on 2nd qubit} = (|010\rangle\langle 010| + |101\rangle\langle 101|) \otimes \mathbb{I}_4 \\ P_3 &= \text{error on 3rd qubit} = (|001\rangle\langle 001| + |110\rangle\langle 110|) \otimes \mathbb{I}_4 \\ P_4 &= \text{error on 4th qubit} = (|000\rangle\langle 000| + |111\rangle\langle 111|) \otimes \mathbb{I}_4 \end{aligned} \tag{15.29}$$

Now we compute the application of P_1 to $|0000\rangle$

$$\begin{aligned} P_1|0000\rangle &= |100\rangle\langle 100||0000\rangle + |011\rangle\langle 011||0000\rangle = \\ &= |100\rangle\underbrace{\langle 100||000\rangle}_{=0}|0\rangle + |011\rangle\underbrace{\langle 011||000\rangle}_{=0}|0\rangle = 0 \end{aligned} \tag{15.30}$$

Notice that only the first three qubit play a role when one applies the projection (on the 4th there is the identity)

So, $P_1|s'_c\rangle = 0$ and analogously $P_2|s'_c\rangle = 0$. Omitting the terms equal to 0 and some other calculations, the application of P_3 and P_4 gives, with an example of application of P_3 on $|0011\rangle$:

$$\begin{aligned} P_3|0011\rangle &= |001\rangle\langle 001||0011\rangle + |110\rangle\langle 110||0011\rangle = \\ &= |001\rangle\underbrace{\langle 001||001\rangle}_{=1}|1\rangle + |110\rangle\langle 110||001\rangle|1\rangle = |001\rangle|1\rangle \end{aligned} \tag{15.31}$$

$$\begin{aligned} P_3|s'_c\rangle &= \frac{a|001\rangle + b|110\rangle}{\sqrt{2}} \otimes |1\rangle \rightarrow X_3 \rightarrow |s_c\rangle \otimes |1\rangle \\ P_4|s'_c\rangle &= \frac{a|000\rangle + b|111\rangle}{\sqrt{2}} \otimes |0\rangle \rightarrow \mathbb{I} \rightarrow |s_c\rangle \otimes |0\rangle \end{aligned} \tag{15.32}$$

16. Shor's Algorithm

The aim is to factor a natural number in polynomial (in the number of bits) time. In RSA it takes an exponential amount of time to find the factors of $N = pq$, with p and q primes, knowing N.

Shor's algorithm finds the order of some $a \in \mathbb{Z}_N$ prime with respect to N.

Exercise 16.1: Factor the number $M = 39$

Choose $a = 5$, $\gcd(a, M) = \gcd(5, 39) = 1$ (a and M are relatively prime)

N.B.: It is fast to compute gcd and then to decide whether two numbers are relatively prime or not, through Euler's algorithm.

Step 1: Compute the order of 5 in $\mathbb{Z}_{39}$ or with respect to 39.

The order of a with respect to M is the smallest number r such that

$$a^r = 1 \bmod M \tag{16.1}$$

$5^0 = 1,\ 5^1 = 5,\ 5^2 = 25,\ 5^3 = 125 = 8 \bmod 39,\ 5^4 = 40 \equiv 1 \bmod 39$

So the order $r = 4$

Step 2: Factor $a^r - 1$

$k \cdot 39 = 0 \bmod 39 = 5^4 - 1 = (5^2 + 1)(5^2 - 1) = 24 \cdot 26$

> *Remark*: From the identity $26 \cdot 24 = k \cdot 39$ all the prime factors of 39 are contained in $26 \cdot 24$ so they are present either in 24 or 26.
>
> In order to find such prime divisors, instead of decomposing 24 and 26 (which is in general long) we can proceed by the following step (*Step 3*)

Step 3: Computing the $\gcd\big(a^{r/2} - 1, M\big)$ and $\gcd\big(a^{r/2} + 1, M\big)$

$\gcd(24, 39) = 3, \quad \gcd(26, 39) = 13$

As a result, $39 = 3 \cdot 13$

Example:

$$\begin{aligned} 2^4 = 1 \bmod 15 &\Rightarrow (2^2 - 1)(2^2 + 1) = k \cdot 15 \\ 2^8 = 1 \bmod 15 &\Rightarrow (2^4 - 1)(2^4 + 1) = k \cdot 15 \end{aligned} \tag{16.2}$$

N.B.: From this identity one does not get the factorization of 15

Example:

$M = 26, \quad a = 3, \quad \Rightarrow \quad 3^0 = 1, 3^1 = 3, 3^2 = 9, 3^3 = 27 = 26 + 1 \equiv 1 \bmod 26$

Thus $r = 3$ and following the previous steps $\big(3^{3/2} - 1\big)\big(3^{3/2} + 1\big)$ but it is not allowed in $\mathbb{Z}_{26}$

The method works for all cases in which:

- the order is even
- the factors of M are not all in either $a^{r/2}-1$ or $a^{r/2}+1$

16.1. A bit of theory

1. Existence of the order
2. Choosing a randomly gives a probability larger than $\frac{1}{2}$ to succeed in passing from the order of a to the factor of M

Proof: Existence of the order

Given M and $0 < a \leq M-1, \quad \gcd(a,M)=1$. This means that a^x is periodic in x and $\exists \bar{x}$ s.t. $a^x \equiv 1 \bmod M$

$a^x \in \{1,...,M-1\}$ which is a finite set, while $x \in \mathbb{N}$ then

$$\exists\ x \text{ and } y, x<y \text{ s.t. } a^x = a^y = a^{x+T} \quad \text{where } T=y-x>0 \tag{16.3}$$

Now $\gcd(a,M)=1 \Rightarrow \gcd(a^x,M)=1 \Rightarrow \exists b \in \mathbb{Z}_M \text{ s.t. } ba^x=1$

Therefore,

$$ba^x = ba^y = ba^{x+T} = ba^x a^T \Rightarrow a^T = 1 \tag{16.4}$$

Moreover, $\forall x \in \mathbb{N}$,

$$a^x = a^x 1 = a^x a^T = a^{x+T} \Rightarrow a^x \text{ is periodic in } x \tag{16.5}$$

□

Corollary 14.1: It exists a minimal period, which is also the minimal number r such that:

$$a^r = 1 \bmod M \tag{16.6}$$

This is called the **order** of a

Remark: The fact that M is not prime is used in the method we developed in the identity $\left(a^{r/2}-1\right)\left(a^{r/2}+1\right) \equiv 0 \bmod M$.

Since this is fulfilled by both non-trivial factors, this implies that $\mathbb{Z}_M$ is not an integrity domain, which is equivalent to saying that M is not prime.

16.1.1. The choice of a:

To factor M, pick $a \in \{2,...,M-1\}$ and compute the order r of a

$$\begin{gathered} a^r = 1 \bmod M \\ \left(a^{\frac{r}{2}}-1\right)\left(a^{\frac{r}{2}}+1\right) = 0 \bmod M \\ \gcd\left(a^{\frac{r}{2}}-1, M\right) \text{ and } \quad \gcd\left(a^{\frac{r}{2}}+1, M\right) \text{ are divisors of } \quad M \end{gathered} \tag{16.7}$$

a is to be chosen such that:

1. r is even.

2. $a^{\frac{r}{2}} - 1$ and $a^{\frac{r}{2}} + 1$ are not zero.

Example: factor $M = 77$

a has to be coprime with respect to 77 $(77 = 11 \cdot 7)$
Lemma: Let r_1 be the order of a with respect to 7, r_2 the order of a with respect to 11, then

$$r \triangleq \operatorname{lcm}(r_1, r_2) \tag{16.8}$$

Proof: r must be a multiple of r_1 and r_2. Indeed:

$$\begin{gathered} a^r = 1 \bmod 77 \Rightarrow a^r = 1 + 77k = 1 + 7 \cdot 11k \\ a^r = 1 \bmod 7 \text{ and } a^r = 1 \bmod 11 \end{gathered} \tag{16.9}$$

Now, $a^r = 1 \bmod 7$ implies that r is a multiple of r_1. If not,

$$1 \equiv a^r = a^{[r/r_1]r_1} a^{r'} \tag{16.10}$$

where $[\cdot]$ denotes the integer part and r' is the remainder of the division $\frac{r}{r_1}$, so $r' < r_1$ and starting from Equation (16.10):

$$1 \equiv 1^{[r/r_1]} a^{r'} \Rightarrow a^{r'} = 1 \bmod 7 \tag{16.11}$$

Which is impossible since $r' < r_1$. r' cannot be the order of $a \bmod 7$, i.e. the least number ρ for which $a^\rho = 1$.

Analogously, r is a multiple of r_2. Now we need to prove that if ρ is a multiple of r_1 and r_2, then $a^\rho = 1 \bmod 77$.
Indeed, $\rho = k_1 r_1 = k_2 r_2$. Then,

$$\begin{gathered} a^\rho = a^{k_1 r_1} \equiv 1^{k_1} \bmod 7 = 1 \bmod 7 = 1 + j_1 \cdot 7 \\ a^\rho = a^{k_2 r_2} \equiv 1^{k_2} \bmod 11 = 1 \bmod 11 = 1 + j_2 \cdot 11 \\ j_1 \cdot 7 = j_2 \cdot 11 = j \cdot 7 \cdot 11 \\ a^\rho = 1 + j \cdot 77 \equiv 1 \bmod 77 \end{gathered} \tag{16.12}$$

Finally $a^\rho \equiv 1 \bmod 77$ iff ρ is a multiple of r_1 and r_2. The order r must be the $\operatorname{lcm}(r_1, r_2)$. □

In conclusion, to single out the elements a that have odd order, one has to study the order of the elements of $\mathbb{Z}_7^*$ and $\mathbb{Z}_{11}^*$

Lemma: The problem

$$\begin{aligned} x &\equiv a_1 \bmod 7 \\ x &\equiv a_2 \bmod 11 \end{aligned} \tag{16.13}$$

Always admits one solution only.

Proof: Firstly, we prove the existence:

$$\begin{aligned} x \equiv a_1 \bmod 7 &\Rightarrow x = a_1 + 11k_2 \\ x \equiv a_2 \bmod 11 &\Rightarrow x = a_1 + 7k_1 \\ &\Rightarrow a_1 + 7k_1 = a_2 + 11k_2 \\ &\Rightarrow a_1 - a_2 = 11k_2 - 7k_1 \end{aligned} \tag{16.14}$$

The problem is to find a couple k_1, k_2 that fulfills the relation.

We can see that choosing $k_2 = 2(a_1 - a_2)$ and $k_1 = 3(a_1 - a_2)$ it is fulfilled, which means:

$$\begin{aligned} x &= a_1 + 3 \cdot 7(a_1 - a_2) = 22a_1 - 21a_2 \\ x &= a_2 + 2 \cdot 11(a_1 - a_2) = 22a_1 - 21a_2 \end{aligned} \tag{16.15}$$

Now we have to prove uniqueness. Suppose to have two such solutions:

$$\begin{aligned} x &= a_1 + 7 \cdot k_1 = a_2 + 11 \cdot k_2 \\ y &= a_1 + 7 \cdot k_1' = a_2 + 11 \cdot k_2' \\ x - y &= 7(k_1 - k_1') = 11(k_2 - k_2') = 5 \cdot 77 \Rightarrow x \equiv y \bmod 77 \end{aligned} \tag{16.16}$$

□

Then one has to exclude 8 possible a's from the initial choice, that was of 60 elements.

Remark: instead of 77, one can choose another number with two prime factors p and q. The final result is a corollary of the *Chinese Remainder Theorem*!

16.1.2. Second and last problem

We want to find the a's for which r is even but

$$\left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right) \equiv 0 \bmod M (= 77) \tag{16.17}$$

but

$$\gcd\left(a^{\frac{r}{2}} - 1, M\right) = M \text{ or } \gcd\left(a^{\frac{r}{2}} + 1, M\right) = M \tag{16.18}$$

and this is impossible. If

$$\begin{aligned} \gcd\left(a^{\frac{r}{2}} - 1, M\right) = M &\Rightarrow a^{\frac{r}{2}} - 1 = kM \\ a^{\frac{r}{2}} &\equiv 1 \bmod M \end{aligned} \tag{16.19}$$

But $\frac{r}{2} < r = \min\{\rho \text{ s.t. } a^\rho \equiv 1 \bmod M\}$ is **absurd**. So, the only possible occurrence to discuss is

$$\gcd\left(a^{\frac{r}{2}} + 1, M\right) = M \Rightarrow a^{\frac{r}{2}} \equiv -1 \bmod M \tag{16.20}$$

Lemma: If $b \equiv -1 \bmod 77$, then $b \equiv -1 \bmod 7$ and $b \equiv -1 \bmod 11$

Proof:

$$b \equiv -1 \bmod 77 \Leftrightarrow b = -1 + k \cdot 77 = -1 + k_1 \cdot 7 = -1 + k_2 \cdot 11 \tag{16.21}$$

So $b \equiv -1 \bmod 7$ and $b \equiv -1 \bmod 11$ □

We have that $a^{\frac{r}{2}} \equiv -1 \bmod 7$ and $b \equiv -1 \bmod 11$. Now, which element of $\mathbb{Z}_7^*$ and of $\mathbb{Z}_{11}^*$ can a be? Only an element whose power is -1.

After looking at the respective multiplicative tables of $\mathbb{Z}_7^*$ and $\mathbb{Z}_{11}^*$, a can be:

$$a \equiv \begin{cases} 3 \text{ or } 5 \text{ or } 6 \text{ in } \mathbb{Z}_7^* \\ 2 \text{ or } 6 \text{ or } 7 \text{ or } 8 \text{ or } 10 \text{ in } \mathbb{Z}_{11}^* \end{cases} \tag{16.22}$$

They are 15 elements. We have to exclude **at most** 15 elements, plus 8 from the previous observations, obtaining 23. We prove that this result is optimal. This means that out of 60, 37 are good.

Lemma: If $x = b \bmod 7$ and $x = b \bmod 11 \Rightarrow x = b \bmod 77$

Proof:

$$x = b + 7 \cdot k_1 = b + 11 \cdot k_2 \Rightarrow 7k_1 = 11k_2 = 77k \tag{16.23}$$

□

So, in the case we analyzed throughout this topic, $a^{15} = -1 \bmod 77$

Example: consider the case :

$$\begin{aligned} a \equiv 3 \bmod 7 \text{ and } a \equiv 10 \bmod 11 \\ r_1 = 6,\ r_2 = 2 \\ \Rightarrow a^3 \equiv -1 \bmod 7 \\ a^3 \equiv -1 \bmod 11 \\ \Rightarrow a^3 \equiv -1 \bmod 77 \end{aligned} \tag{16.24}$$

17. Help

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

0	0	**1**	1	**2**	2	**3**	3	**4**	4	**5**	5	**6**	6	**7**	7	**8**	8	**9**	9	**10**	10	**11**	11	**12**	12	**13**	13	**14**	14	**15**	15	**16**	16	**17**	17	**18**	18	**19**	19
20	20	**21**	21	**22**	22	**23**	23	**24**	24	**25**	25	**26**	0	**27**	1	**28**	2	**29**	3	**30**	4	**31**	5	**32**	6	**33**	7	**34**	8	**35**	9	**36**	10	**37**	11	**38**	12	**39**	13
40	14	**41**	15	**42**	16	**43**	17	**44**	18	**45**	19	**46**	20	**47**	21	**48**	22	**49**	23	**50**	24	**51**	25	**52**	0	**53**	1	**54**	2	**55**	3	**56**	4	**57**	5	**58**	6	**59**	7
60	8	**61**	9	**62**	10	**63**	11	**64**	12	**65**	13	**66**	14	**67**	15	**68**	16	**69**	17	**70**	18	**71**	19	**72**	20	**73**	21	**74**	22	**75**	23	**76**	24	**77**	25	**78**	0	**79**	1
80	2	**81**	3	**82**	4	**83**	5	**84**	6	**85**	7	**86**	8	**87**	9	**88**	10	**89**	11	**90**	12	**91**	13	**92**	14	**93**	15	**94**	16	**95**	17	**96**	18	**97**	19	**98**	20	**99**	21
100	22	**101**	23	**102**	24	**103**	25	**104**	0	**105**	1	**106**	2	**107**	3	**108**	4	**109**	5	**110**	6	**111**	7	**112**	8	**113**	9	**114**	10	**115**	11	**116**	12	**117**	13	**118**	14	**119**	15
120	16	**121**	17	**122**	18	**123**	19	**124**	20	**125**	21	**126**	22	**127**	23	**128**	24	**129**	25	**130**	0	**131**	1	**132**	2	**133**	3	**134**	4	**135**	5	**136**	6	**137**	7	**138**	8	**139**	9
140	10	**141**	11	**142**	12	**143**	13	**144**	14	**145**	15	**146**	16	**147**	17	**148**	18	**149**	19	**150**	20	**151**	21	**152**	22	**153**	23	**154**	24	**155**	25	**156**	0	**157**	1	**158**	2	**159**	3
160	4	**161**	5	**162**	6	**163**	7	**164**	8	**165**	9	**166**	10	**167**	11	**168**	12	**169**	13	**170**	14	**171**	15	**172**	16	**173**	17	**174**	18	**175**	19	**176**	20	**177**	21	**178**	22	**179**	23
180	24	**181**	25	**182**	0	**183**	1	**184**	2	**185**	3	**186**	4	**187**	5	**188**	6	**189**	7	**190**	8	**191**	9	**192**	10	**193**	11	**194**	12	**195**	13	**196**	14	**197**	15	**198**	16	**199**	17
200	18	**201**	19	**202**	20	**203**	21	**204**	22	**205**	23	**206**	24	**207**	25	**208**	0	**209**	1	**210**	2	**211**	3	**212**	4	**213**	5	**214**	6	**215**	7	**216**	8	**217**	9	**218**	10	**219**	11
220	12	**221**	13	**222**	14	**223**	15	**224**	16	**225**	17	**226**	18	**227**	19	**228**	20	**229**	21	**230**	22	**231**	23	**232**	24	**233**	25	**234**	0	**235**	1	**236**	2	**237**	3	**238**	4	**239**	5
240	6	**241**	7	**242**	8	**243**	9	**244**	10	**245**	11	**246**	12	**247**	13	**248**	14	**249**	15	**250**	16	**251**	17	**252**	18	**253**	19	**254**	20	**255**	21	**256**	22	**257**	23	**258**	24	**259**	25
260	0	**261**	1	**262**	2	**263**	3	**264**	4	**265**	5	**266**	6	**267**	7	**268**	8	**269**	9	**270**	10	**271**	11	**272**	12	**273**	13	**274**	14	**275**	15	**276**	16	**277**	17	**278**	18	**279**	19
280	20	**281**	21	**282**	22	**283**	23	**284**	24	**285**	25	**286**	0	**287**	1	**288**	2	**289**	3	**290**	4	**291**	5	**292**	6	**293**	7	**294**	8	**295**	9	**296**	10	**297**	11	**298**	12	**299**	13
300	14	**301**	15	**302**	16	**303**	17	**304**	18	**305**	19	**306**	20	**307**	21	**308**	22	**309**	23	**310**	24	**311**	25	**312**	0	**313**	1	**314**	2	**315**	3	**316**	4	**317**	5	**318**	6	**319**	7
320	8	**321**	9	**322**	10	**323**	11	**324**	12	**325**	13	**326**	14	**327**	15	**328**	16	**329**	17	**330**	18	**331**	19	**332**	20	**333**	21	**334**	22	**335**	23	**336**	24	**337**	25	**338**	0	**339**	1
340	2	**341**	3	**342**	4	**343**	5	**344**	6	**345**	7	**346**	8	**347**	9	**348**	10	**349**	11	**350**	12	**351**	13	**352**	14	**353**	15	**354**	16	**355**	17	**356**	18	**357**	19	**358**	20	**359**	21
360	22	**361**	23	**362**	24	**363**	25	**364**	0	**365**	1	**366**	2	**367**	3	**368**	4	**369**	5	**370**	6	**371**	7	**372**	8	**373**	9	**374**	10	**375**	11	**376**	12	**377**	13	**378**	14	**379**	15
380	16	**381**	17	**382**	18	**383**	19	**384**	20	**385**	21	**386**	22	**387**	23	**388**	24	**389**	25	**390**	0	**391**	1	**392**	2	**393**	3	**394**	4	**395**	5	**396**	6	**397**	7	**398**	8	**399**	9
400	10	**401**	11	**402**	12	**403**	13	**404**	14	**405**	15	**406**	16	**407**	17	**408**	18	**409**	19	**410**	20	**411**	21	**412**	22	**413**	23	**414**	24	**415**	25	**416**	0	**417**	1	**418**	2	**419**	3
420	4	**421**	5	**422**	6	**423**	7	**424**	8	**425**	9	**426**	10	**427**	11	**428**	12	**429**	13	**430**	14	**431**	15	**432**	16	**433**	17	**434**	18	**435**	19	**436**	20	**437**	21	**438**	22	**439**	23
440	24	**441**	25	**442**	0	**443**	1	**444**	2	**445**	3	**446**	4	**447**	5	**448**	6	**449**	7	**450**	8	**451**	9	**452**	10	**453**	11	**454**	12	**455**	13	**456**	14	**457**	15	**458**	16	**459**	17
460	18	**461**	19	**462**	20	**463**	21	**464**	22	**465**	23	**466**	24	**467**	25	**468**	0	**469**	1	**470**	2	**471**	3	**472**	4	**473**	5	**474**	6	**475**	7	**476**	8	**477**	9	**478**	10	**479**	11
480	12	**481**	13	**482**	14	**483**	15	**484**	16	**485**	17	**486**	18	**487**	19	**488**	20	**489**	21	**490**	22	**491**	23	**492**	24	**493**	25	**494**	0	**495**	1	**496**	2	**497**	3	**498**	4	**499**	5
500	6	**501**	7	**502**	8	**503**	9	**504**	10	**505**	11	**506**	12	**507**	13	**508**	14	**509**	15	**510**	16	**511**	17	**512**	18	**513**	19	**514**	20	**515**	21	**516**	22	**517**	23	**518**	24	**519**	25
520	0	**521**	1	**522**	2	**523**	3	**524**	4	**525**	5	**526**	6	**527**	7	**528**	8	**529**	9	**530**	10	**531**	11	**532**	12	**533**	13	**534**	14	**535**	15	**536**	16	**537**	17	**538**	18	**539**	19
540	20	**541**	21	**542**	22	**543**	23	**544**	24	**545**	25	**546**	0	**547**	1	**548**	2	**549**	3	**550**	4	**551**	5	**552**	6	**553**	7	**554**	8	**555**	9	**556**	10	**557**	11	**558**	12	**559**	13
560	14	**561**	15	**562**	16	**563**	17	**564**	18	**565**	19	**566**	20	**567**	21	**568**	22	**569**	23	**570**	24	**571**	25	**572**	0	**573**	1	**574**	2	**575**	3	**576**	4	**577**	5	**578**	6	**579**	7
580	8	**581**	9	**582**	10	**583**	11	**584**	12	**585**	13	**586**	14	**587**	15	**588**	16	**589**	17	**590**	18	**591**	19	**592**	20	**593**	21	**594**	22	**595**	23	**596**	24	**597**	25	**598**	0	**599**	1
600	2	**601**	3	**602**	4	**603**	5	**604**	6	**605**	7	**606**	8	**607**	9	**608**	10	**609**	11	**610**	12	**611**	13	**612**	14	**613**	15	**614**	16	**615**	17	**616**	18	**617**	19	**618**	20	**619**	21
620	22	**621**	23	**622**	24	**623**	25	**624**	0	**625**	1	**626**	2	**627**	3	**628**	4	**629**	5	**630**	6	**631**	7	**632**	8	**633**	9	**634**	10	**635**	11	**636**	12	**637**	13	**638**	14	**639**	15
640	16	**641**	17	**642**	18	**643**	19	**644**	20	**645**	21	**646**	22	**647**	23	**648**	24	**649**	25	**650**	0	**651**	1	**652**	2	**653**	3	**654**	4	**655**	5	**656**	6	**657**	7	**658**	8	**659**	9
660	10	**661**	11	**662**	12	**663**	13	**664**	14	**665**	15	**666**	16	**667**	17	**668**	18	**669**	19	**670**	20	**671**	21	**672**	22	**673**	23	**674**	24	**675**	25	**676**	0	**677**	1	**678**	2	**679**	3
680	4	**681**	5	**682**	6	**683**	7	**684**	8	**685**	9	**686**	10	**687**	11	**688**	12	**689**	13	**690**	14	**691**	15	**692**	16	**693**	17	**694**	18	**695**	19	**696**	20	**697**	21	**698**	22	**699**	23

Number	1	3	5	7	9	11	15	17	19	21	23	25
Inverse mod 26	1	9	21	15	3	19	7	23	11	5	17	25

17.1. Irreducible polynomials

(0,1,2)	(0,1,3,4,24)	(0,1,46)	(0,1,5,7,68)	(0,2,3,5,90)	(0,3,4,5,112)	(0,1,3)	(0,3,25)
(0,5,47)	(0,2,5,6,69)	(0,1,5,8,91)	(0,2,3,5,113)	(0,1,4)	(0,1,3,4,26)	(0,2,3,5,48)	(0,1,3,5,70)
(0,2,5,6,92)	(0,2,3,5,114)	(0,2,5)	(0,1,2,5,27)	(0,4,5,6,49)	(0,1,3,5,71)	(0,2,93)	(0,5,7,8,115)
(0,1,6)	(0,1,28)	(0,2,3,4,50)	(0,3,9,10,72)	(0,1,5,6,94)	(0,1,2,4,116)	(0,1,7)	(0,2,29)
(0,1,3,6,51)	(0,2,3,4,73)	(0,11,95)	(0,1,2,5,117)	(0,1,3,4,8)	(0,1,30)	(0,3,52)	(0,1,2,6,74)
(0,6,9,10,96)	(0,2,5,6,118)	(0,1,9)	(0,3,31)	(0,1,2,6,53)	(0,1,3,6,75)	(0,6,97)	(0,8,119)
(0,3,10)	(0,2,3,7,32)	(0,3,6,8,54)	(0,2,4,5,76)	(0,3,4,7,98)	(0,1,3,4,120)	(0,2,11)	(0,1,3,6,33)
(0,1,2,6,55)	(0,2,5,6,77)	(0,1,3,6,99)	(0,1,5,8,121)	(0,3,12)	(0,1,3,4,34)	(0,2,4,7,56)	(0,1,2,7,78)
(0,2,5,6,100)	(0,1,2,6,122)	(0,1,3,4,13)	(0,2,35)	(0,4,57)	(0,2,3,4,79)	(0,1,6,7,101)	(0,2,123)
(0,5,14)	(0,2,4,5,36)	(0,1,5,6,58)	(0,2,4,9,80)	(0,3,5,6,102)	(0,37,124)	(0,1,15)	(0,1,4,6,37)
(0,2,4,7,59)	(0,4,81)	(0,9,103)	(0,5,6,7,125)	(0,1,3,5,16)	(0,1,5,6,38)	(0,1,60)	(0,4,6,9,82)
(0,1,3,4,104)	(0,2,4,7,126)	(0,3,17)	(0,4,39)	(0,1,2,5,61)	(0,2,4,7,83)	(0,4,105)	(0,1,127)
(0,3,18)	(0,3,4,5,40)	(0,3,5,6,62)	(0,5,84)	(0,1,5,6,106)	(0,1,2,7,128)	(0,1,2,5,19)	(0,3,41)
(0,1,63)	(0,1,2,8,85)	(0,4,7,9,107)	(0,3,20)	(0,1,2,5,42)	(0,1,3,4,64)	(0,2,5,6,86)	(0,1,4,6,108)
(0,2,21)	(0,3,4,6,43)	(0,1,3,4,65)	(0,1,5,7,87)	(0,2,4,5,109)	(0,1,22)	(0,5,44)	(0,3,66)
(0,8,9,11,88)	(0,1,4,6,110)	(0,5,23)	(0,1,3,4,45)	(0,1,2,5,67)	(0,3,5,6,89)	(0,2,4,7,111)	