SCIENCES SORBONNE UNIVERSITÉ

M1 SFPN Project

Cold Boot Attack on AES

Amadou Bayo and Hichem Aggoun

Supervised by Charles Bouillaguet and Damien Vergnaud

Figure 1: Cold Boot Attack illustration [7]

Contents

List of Figures

Acknowledgements

To begin, we want to thank Damien Vergnaud and Charles Bouillaguet who proposed this subject. We thank them for all their involvements during this project. In fact they permit us to learn and understand the AES and some communications channel models more deeply.

1 Introduction

Nowadays, a tremendous part of our life, professional or not, reside in all our devices' memory. As strange as this might sound, this information doesn't immediately disappear after turning off our devices. There are numerous communications channel models to simulate the evolution of the information. Cold Boot attacks use this fact. They enable, just with an access to the machine during a couple of minutes, to grab information contained in the RAM. In all the data we can find in the RAM, let's focus on AES (Advanced Encryption Standard) keys.
Cold Boot attack on AES was discovered in 2008 by Halderman et al. [1]. and it is still feasible nowadays, indeed in 2018, Olle Segerdahl and Pasi Saarinen proved it for the most used brands of laptops [3].
They are among the family of attack called side-channel attacks. These attacks exploit the implementation of standards or protocols in a system, the attack is simplified thanks to the implementation. For our case, we focus on AES. Despite AES is unbroken for more than twenty years, it is unfortunately not resistant to side-channel attacks.
To perform this attack, we need to perfectly know how the AES works.
Firstly, we will describe more specifically the Cold Boot Attack. Secondly, we will focus on AES and the main operations on its keys. Finally, we will characterize the different communications channel models and the attacks linked to them.

Program functionalities :
- Production of a decayed AES key schedule according to the binary erasure channel and the Z-channel

- Correction of the Key Schedule, decayed in agreement with the binary erasure channel and retrieving of the master key

- Correction of the Key Schedule, decayed in agreement with the Z-channel and retrieving of the master key

To implement the algorithms presented through the paper, we will use the **Python** programming language in its third version. It facilitates us thanks to its high number of cryptography libraries and its interactivity.

2 Cold Boot Attack

2.1 Remanence in the RAM

As said in the introduction, the cold boot attacks are based on the remanence time of the data in the memory. We know this time exists but it can even be increased using coolants as liquid nitrogen.
In [1], an illustration helps us to understand this :

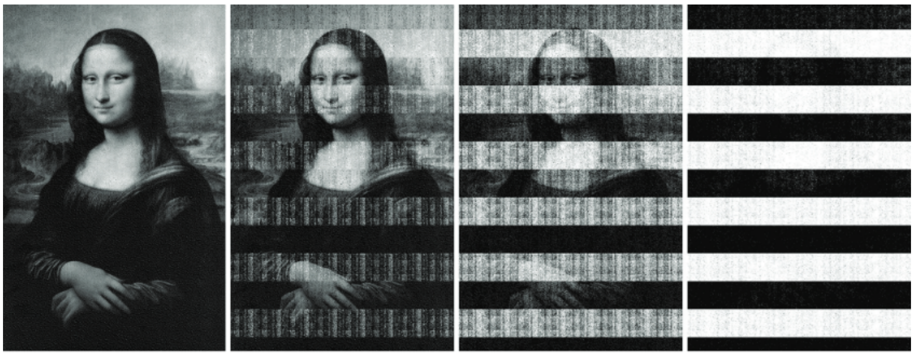


Figure 2: Remanence in the RAM

This shows how information decay within the RAM. The experiment is to load an image into memory and then cut power for varying intervals. For the first image the cut is after 5 seconds, the second image after 30 seconds, the third 60 seconds, and the last one 5 min. Knowing that the chips remained close to room temperature, we see that we still can recognize the original. Using cooling techniques would increase these times.
Another experiment with memory remanence is to fill the RAM of a known string, then force a reboot and finally verify if we can find out the string. If so, we would understand some of the contents of memory survived the reboot.
Knowing that this remanence exists, the process of the attack is the following :

- Have an access to the machine
- Cooling down the RAM chips thanks to nitrogen
- Retrieve the information in the RAM or even the RAM chips directly
- Find the decayed encryption keys within the RAM
- Correct the keys

In our project, we only study the algorithmic part of the attack and more specifically, the reconstruction of the encryption keys (the last step).

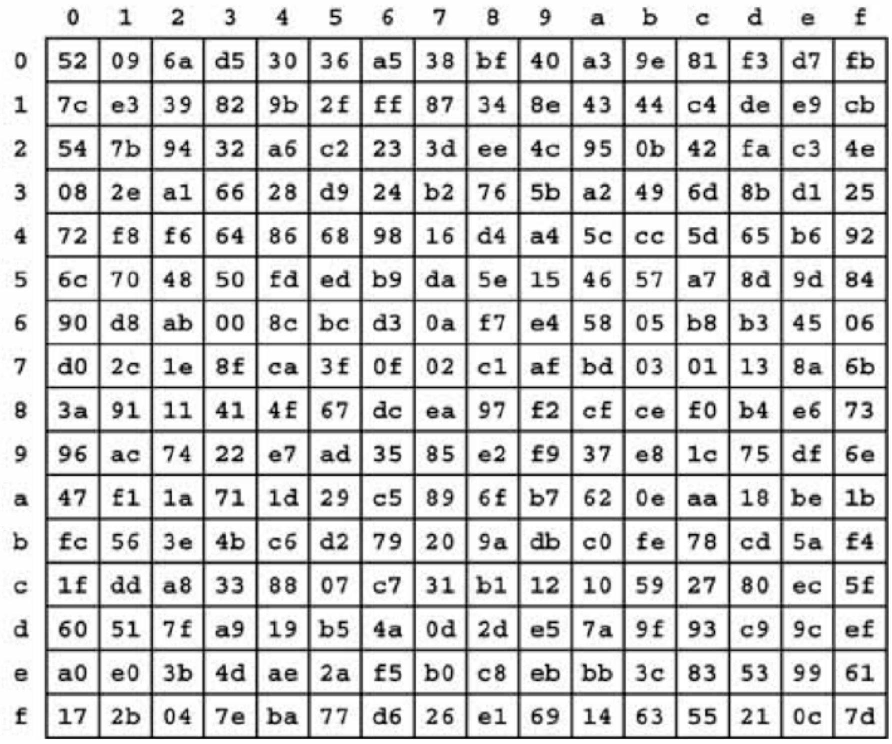
3 Advanced Encryption Standard

3.1 A little description

AES is one the most used Encryption Standard nowadays. It has been created by Joan Daemen and Vincent Rijmen in 1997. It is also called Rjindael encryption because of the names of the creators. AES is formed of an encryption process and a key expansion. As the attack depends on the key expansion, we will limit our description to the necessary operations for the key expansion also called key schedule.

3.2 AES Key Schedule

AES exists for three different lengths of key. We only examine the 128 bit keys. With this length, we have to deal with 11 sub-keys, the first one is the master key and permits to begin the process. One sub-key is created at each round and so there are 10 rounds. Each sub-key is represented by a 4x4 matrix.
First, we will describe two main operations in the AES key schedule : S-box (S is for substitution) and RotWord.
The S-box is an array which substitutes bytes :



	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 3: Rjindael S-box [2]

As instance, b8 would become 9a. We can easily deduce the inverse S-box. All the operations of AES are easily reversing.

RotWord is a one-byte left circular shift : RotWord([a b c d]) = [b c d a]

Finally, RCON is for 'round constant', it is 10 bytes and each of them is linked to a round :

1	2	3	4	5	6	7	8	9	10
01	02	04	08	10	20	40	80	1B	36

Figure 4: Round constant AES [5]

We can now describe the construction of each sub-key, it is built this way :

The first column of the following sub-key is obtained by making a XOR operation between :

- the first column of the current sub-key
- the last column of the current sub-key on which S-box and RotWord were applied
- the corresponding byte from RCON.

Here is an exemple of calculation for the first column of the second sub-key :

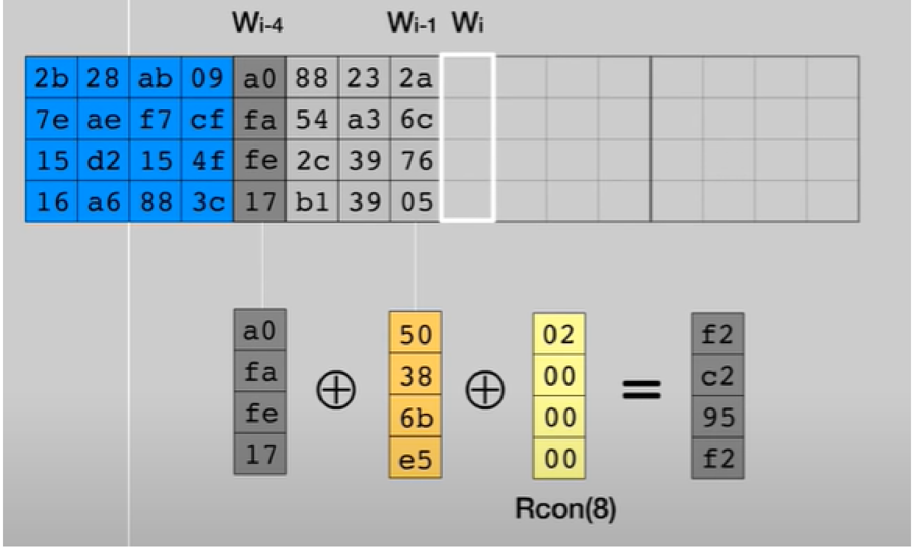


Figure 5: First Column AES sub-key [6]

The three last columns of the sub-key are simply calculated by XORing two columns as we can see in the following illustration :

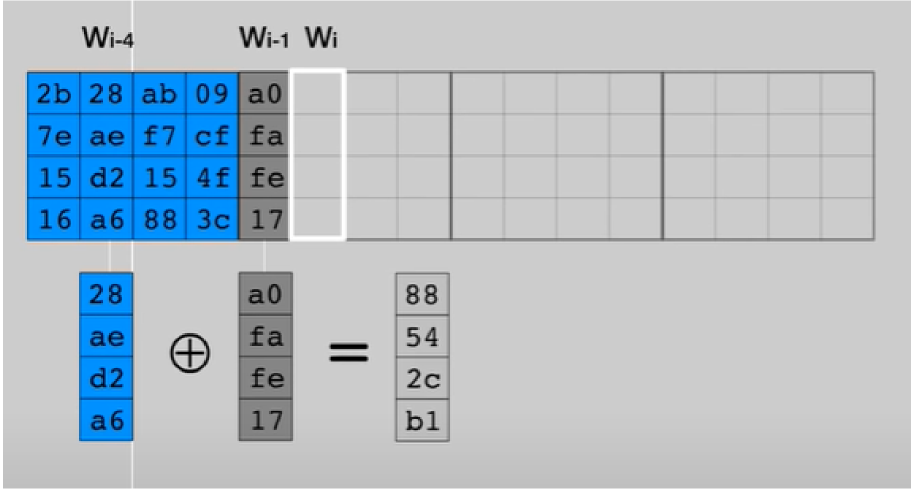


Figure 6: Second Column AES sub-key [6]

All of these operations are very important in order to exploit the AES key schedule. Once the operations of AES are understood, we can examine the communications channel. Notice that each channel has a its own attack on the AES keys.

4 Different communications channel models

4.1 The Binary Erasure Channel

4.1.1 Description of the channel

This channel simulates the fact that some bits can be erased minutes after minutes. This channel is the simplest because we are sure that the bits still readable are correct which is not the case with other channels.

In our implementation of the channel, we directly erase the bytes with a given probability. Thus, the bytes are transformed this way : as instance, a byte 'ff' would become '??'.

Here is an algorithm which returns a decayed key schedule according to the Binary Erasure Channel :

Algorithm 1 Key Schedule Decaying (Binary Erasure Channel)

```
function KeyDecaying($keySchedule, p$)                ▷ p is the probability of erasure
    for each subkeys do
        for each byte do
            rand = randint(1,100)
            if $rand < p$ then
                $byte \leftarrow '??'$
            end if
        end for
    end for
    return $keySchedule$                               ▷ The key Schedule is decayed
end function
```

This channel does not represent the reality but it's a good way to implement the necessary operations of AES to reconstruct a key schedule.

4.1.2 Attack against this channel

The attack consists of trying to calculate all the possible bytes with the bytes that were not erased. If at the end of a round, no new bytes were calculated, the algorithm stops because this means we cannot go further. In this way, we understand that sometimes, the attack can fail. If we don't have enough information, we will calculate some bytes but not enough to rebuild a sub-key. There are two possibilities :

- If we reach a state where there is no more possible calculation and no sub-key without erased byte was found, the attack fails.

- On the contrary, if only one sub-key without erased byte is found, the attacks is passed.

This is a success, in the second case, because we can rebuild all the key schedule thanks to only one sub-key (whatever which one it is) and reach the master key. Indeed, as we said, all the operations of AES are easily reversing.

We had to implement all the operations described before. As we saw in the description of AES, the sub-keys depends on the previous or the following sub-key. This permits

us to double the ways to calculate erased bytes. More generally, if in the bytes at stake, two bytes on three are known and one of these is erased, we can easily isolate this one and calculate it.
The dependencies of the bytes of the first columns show all the possibilities of calculations :

- They can be calculated using the fact that the second columns needs the first one to be calculated, and so we can isolate it.
- They can be calculated using the first column of the following sub-key and the last column of the current sub-key (on which S-box and RotWord were applied)
- They can also be calculated using the second way but 'in the other direction', using the previous sub-key instead of the following.

Here is the algorithm :

Algorithm 2 Binary Erasure Channel Attack

```
function ERASUREATTACK($KS$)                    ▷ KS the decayed key schedule
    continue ← $true$
    while continue do
        $continue \leftarrow false$
        for each byte in KS do
            Try to calculate this byte
            if the byte is calculated then
                $continue \leftarrow true$
            end if
            if no error subkey is found then
                $sol \leftarrow noErrorSubkey$
                return keySchedule(sol)
            end if
        end for
    end while
end function
```

4.1.3 Results

Our first idea was to exploit the less error sub-key and even the sub-key which follows the less error sub-key but this did not give great results.
As we can see, the way we described in the last paragraph, permits to correct key schedules with 80% of erasure rate :

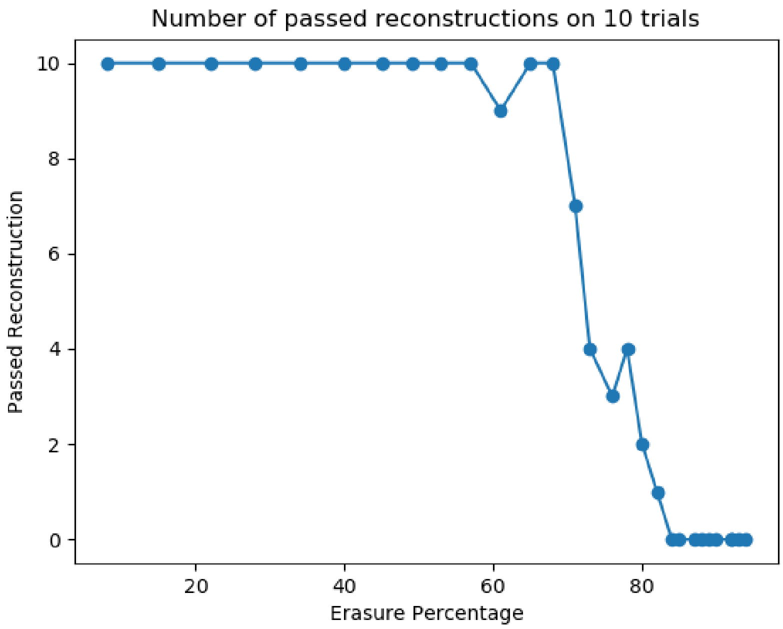


Figure 7: Key schedule reconstruction

Under 65% of erasure percentage, the attack doesn't fail and upper 85%, the reconstruction is totally impossible because there are not enough clean bytes to make any operation. Let's notice that for this channel, the computations are very fast and so the corrected key schedule is found almost instantly.
In some executions, the algorithm failed whereas there was a sub-key with only one byte erased. This byte was impossible to calculate with AES' operations. Therefore we add a brute-force part :
- We try all the possible value for the erased byte (00 to ff)
- For each value, we make the key schedule
- We calculate the hamming distance between the key schedule we just obtained and the key schedule with the erased bytes
- We keep the key schedule which produces the least hamming distance.
We could even make it for more than one byte but the time of execution would increase.

4.2 The Z-Channel (Binary asymmetric channel)

4.2.1 Description of the channel

This channel simulates the fact that when a bit is 0, it cannot change. But when a bit is 1, there is a probability that this can become a 0 in the near future.

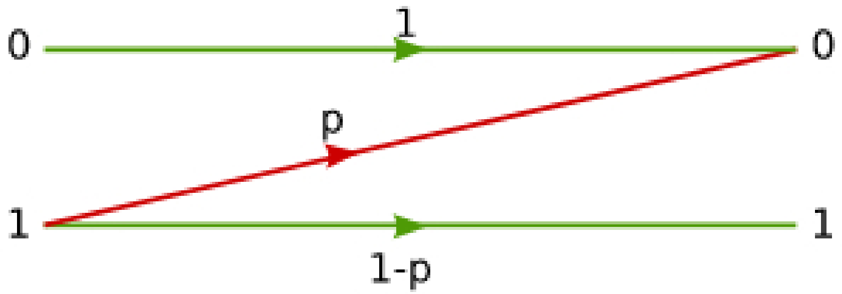


Figure 8: Z-Channel [4]

An explanation is that a bit 1 can loose charge and so become a 0. This channel is harder because there are some bits which are not correct at all. This channel is the one described in the paper and it is closer to the reality.
Here is an algorithm which returns a decayed key schedule according to the Z-Channel :

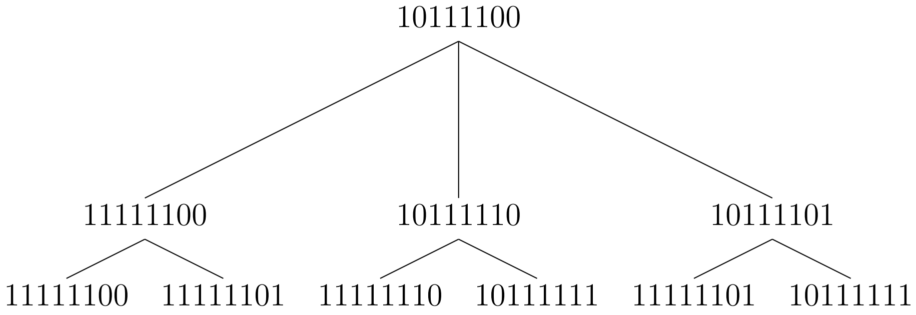
Algorithm 3 Key Schedule Decaying (Z-Channel)

```
function KeyDecaying($keySchedule, p$)        ▷ p is the probability of erasure
    for each subkeys do
        for each bit do
            rand = randint(1,100)
            if $rand < p$ and $bit = 1$ then
                $bit \leftarrow 0$
            end if
        end for
    end for
    return $keySchedule$                       ▷ The key Schedule is decayed
end function
```

As we can see, for this channel, the keys are modified by decaying the bits directly.

4.2.2 Attack against this channel

For this channel, we only study the decayed master key (the first sub-key) and the first column of the following sub-key.
To begin, the goal is to prepare a list of all the potential keys which could have led to the studied sub-key. This step can be done in 2^{128} operations in the worst case as it is the key length, it is a substantial step. As we know in the Z-channel, the '1' can become '0' and we will use it. We will use a tree which present all the possible values that can take a byte according to the Z-channel. Here is an exemple :



After this, we get all the nodes by removing the duplicates and we have all the bytes that could have decayed into the wanted byte. This tree structure shows that the complexity could blow up.

Once we have the combinations for a byte, we have to count the total possible combinations for a column. As it is a 4x4 matrix, the column is made of 4 bytes.
This research of combinations has a complexity of 2^{32} operations in the worst case as we have 4 columns :

Input : $byte_1, byte_2, byte_3, byte_4$
Output : All the combinations of the 4 bytes

Algorithm 4 Columns' Combinations

```
function COMBINATIONS($byte_1, byte_2, byte_3, byte_4$)
    $columns \leftarrow init()$
    for a $taking\ all\ possible\ values\ of\ byte_1$ do
        for b $taking\ all\ possible\ values\ of\ byte_2$ do
            for c $taking\ all\ possible\ values\ of\ byte_3$ do
                for d $taking\ all\ possible\ values\ of\ byte_4$ do
                    add(columns, a + b + c + d)
                end for
            end for
        end for
    end for
end function
```

Finally, the main part of the attack. As described in AES, the first and the last column are linked to compute the first column of the following sub-key. Hence, we will decrease the complexity thanks to the following algorithm which is the same as the last algorithm but on the column instead of the bytes.

Input : $column_1, column_2, column_3, column_4$, decayed key schedule
Output : Corrected Key schedule

Algorithm 5 Z-channel attack

```
function ATTACK($col_1, col_2, col_3, col_4, KS$)        ▷ KS is the decayed key schedule
    for $c1$ in $col_1$ do
        for $c2$ in $col_2$ do
            for $c3$ in $col_3$ do
                for $c4$ in $col_4$ do
                    masterKey = c1 + c2 + c3 + c4
                    g = keySchedule(masterKey)
                    if g was decayed into KS then
                        return $g$        ▷ $g$ is the rectified key Schedule
                    end if
                end for
            end for
        end for
    end for
end function
```

4.2.3 Results

As we can see, the way we described in the last paragraph, permits to correct key schedules with 5% of erasure rate :

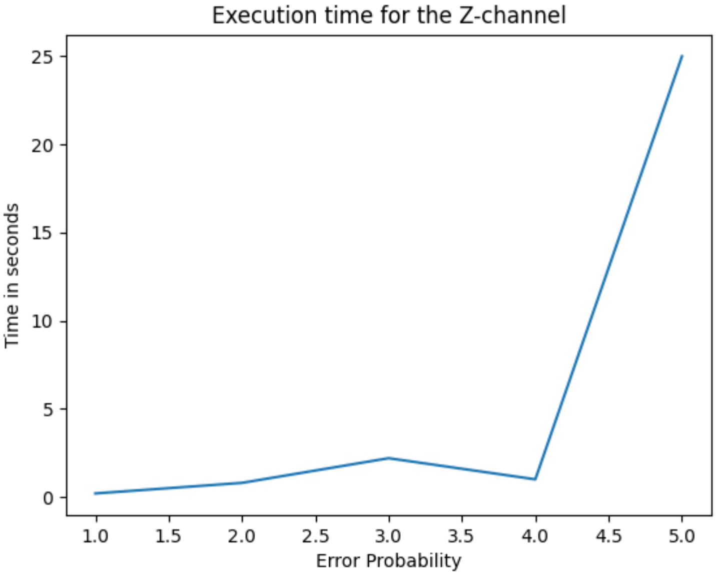


Figure 9: Z-channel performance

It is not sufficient in terms of performance but this implements the right calculations. This method should reach at least 15% of erasure rate in a few seconds.

5 Conclusion

To conclude, we can say that we reach great performances for the Binary Erasure Channel and this permitted us to understand well the AES key expansion. The implementation for the Z-channel is correct but it is improvable seen the bad performance.

About the attack, we found some ways [3] to protect ourselves against this threat. F-Secure gives some tips : for example, it is a good idea to make it compulsory to ask PIN code when restoring or switching on the computer. Of course, keeping laptops in a safe place reduce the risks.

For [1], the most effective way for users to protect themselves is to fully shut down their computers several minutes before any situation and the best software-only solution would be to encrypt the disk key with a password whenever the computer enters an inactive state, so that it will not be useful to an attacker even if it is copied from RAM. Knowing that even when you lock, suspend, or hibernate your computer, the contents of RAM may be preservedâeither in RAM itself or elsewhereâand.

To us, this project allowed us to read some research paper and to learn to find the right information. More technically, as the AES is very used, it is a good thing to have a proper understanding of this standard and this project gave us this comprehension in particular thanks to implementations.

References

[1] J. A. Halderman et al. “Lest we remember: cold-boot attacks on encryption keys”. In: USENIX Security Symposium (2008), pp. 45–60.

[2] Edwin Arboleda et al. “Enhanced Hybrid Algorithm of Secure and Fast Chaos-based, AES, RSA and ElGamal Cryptosystems”. In: Indian Journal of Science and Technology 10.27 (2017), pp. 1–14. DOI: 10.17485/ijst/2017/v10i27/105001.

[3] Olle Segerdahl et al. The Chilling Reality of Cold Boot Attacks. URL: https://blog.f-secure.com/cold-boot-attacks/. (accessed: 24.05.2021).

[4] Benjamin D. Esham. Z-channel. URL: https://en.wikipedia.org/wiki/Z-channel_(information_theory). (accessed: 26.05.2021).

[5] Gerardus T. M. Hubert. Round key generation for aes rijndael block cipher. URL: https://patents.google.com/patent/WO2004002057A2/en. (accessed: 26.05.2021).

[6] Busan Hyeong. AES(Advanced Encryption Standard). URL: https://k-owl.tistory.com/216. (accessed: 26.05.2021).

[7] Security Ninja. Cold Boot Attack. URL: https://resources.infosecinstitute.com/topic/cold-boot-attack/. (accessed: 24.05.2021).