University of Sheffield

Assignment 1: Cryptographic Analysis Task

Jagpreet

Fundamental Security Properties and Mechanisms

Department of Computer Science

December 16, 2022

Contents

List of Figures

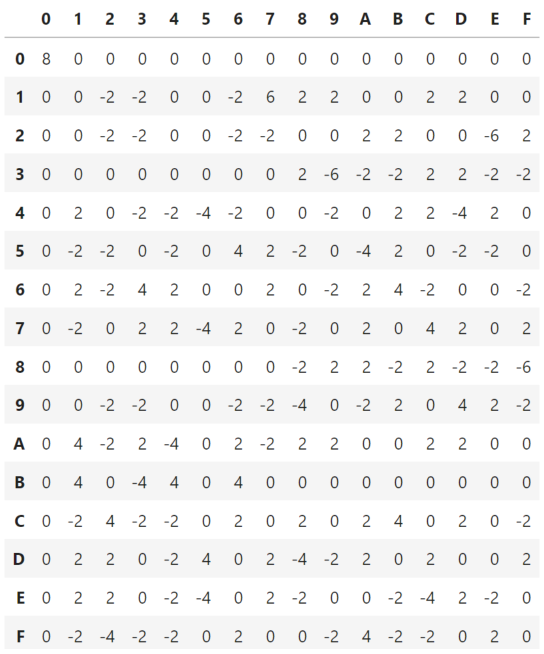
List of Tables

Chapter 1

Question 1. Substitution Permutation Network Cipher and Differential Cryptanalysis

In the First approximation:

1. We begin by making a Linear Approximation Table based on the Substitution box provided, which is implemented as :



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	-2	-2	0	0	-2	6	2	2	0	0	2	2	0	0
2	0	0	-2	-2	0	0	-2	-2	0	0	2	2	0	0	-6	2
3	0	0	0	0	0	0	0	0	2	-6	-2	-2	2	2	-2	-2
4	0	2	0	-2	-2	-4	-2	0	0	-2	0	2	2	-4	2	0
5	0	-2	-2	0	-2	0	4	2	-2	0	-4	2	0	-2	-2	0
6	0	2	-2	4	2	0	0	2	0	-2	2	4	-2	0	0	-2
7	0	-2	0	2	2	-4	2	0	-2	0	2	0	4	2	0	2
8	0	0	0	0	0	0	0	0	-2	2	2	-2	2	-2	-2	-6
9	0	0	-2	-2	0	0	-2	-2	-4	0	-2	2	0	4	2	-2
A	0	4	-2	2	-4	0	2	-2	2	2	0	0	2	2	0	0
B	0	4	0	-4	4	0	4	0	0	0	0	0	0	0	0	0
C	0	-2	4	-2	-2	0	2	0	2	0	2	4	0	2	0	-2
D	0	2	2	0	-2	4	0	2	-4	-2	2	0	2	0	0	2
E	0	2	2	0	-2	-4	0	2	-2	0	0	-2	-4	2	-2	0
F	0	-2	-4	-2	-2	0	2	0	0	-2	4	-2	-2	0	2	0

Figure 1.1: *Approx. Table*

2. A 4-bit number (0111) is taken as input to $S - box_1$

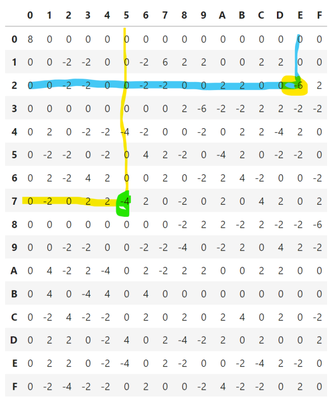
3. Active S-boxes are :

   - $S - box_1$:

$$X_2 \oplus X_3 \oplus X_4 = Y_2 \oplus Y_4, Probability(p) = (8-4)/16 = 4/16 = 1/4 Bias = p - 1/2 = -0.25$$

   - $S - box_3$:

$$X_2 = Y_1 \oplus Y_2 \oplus Y_3 Probability(p) = (8-6)/16 = 2/16 = 1/8 Bias = ((p) - 1/2) = -3/8 = -0.375$$



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	-2	-2	0	0	-2	6	2	2	0	0	2	2	0	0
2	0	0	-2	-2	0	0	-2	-2	0	0	2	2	0	0	-6	2
3	0	0	0	0	0	0	0	0	2	-6	-2	-2	2	2	-2	-2
4	0	2	0	-2	-2	-4	-2	0	0	-2	0	2	2	-4	2	0
5	0	-2	-2	0	-2	0	4	2	-2	0	-4	2	0	-2	-2	0
6	0	2	-2	4	2	0	0	2	0	-2	2	4	-2	0	0	-2
7	0	-2	0	2	2	-4	2	0	-2	0	2	0	4	2	0	2
8	0	0	0	0	0	0	0	0	-2	2	2	-2	2	-2	-2	-6
9	0	0	-2	-2	0	0	-2	-2	-4	0	-2	2	0	4	2	-2
A	0	4	-2	2	-4	0	2	-2	2	2	0	0	2	2	0	0
B	0	4	0	-4	4	0	4	0	0	0	0	0	0	0	0	0
C	0	-2	4	-2	-2	0	2	0	2	0	2	4	0	2	0	-2
D	0	2	2	0	-2	4	0	2	-4	-2	2	0	2	0	0	2
E	0	2	2	0	-2	-4	0	2	-2	0	0	-2	-4	2	-2	0
F	0	-2	-4	-2	-2	0	2	0	0	-2	4	-2	-2	0	2	0

Figure 1.2: *First Approx. table*

- $S - box_4$:

$$X_2 = Y_1 \oplus Y_2 \oplus Y_3, Probability = 2/16 = 1/8 Bias = 1/8 - 1/2 = -0.375$$

4. Combining the equations :

$$U_{31} \oplus U_{32} \oplus U_{33} \oplus U_{34} \oplus U_{35} \oplus U_{36} \oplus P_2 \oplus P_3 \oplus P_4 \oplus \sum K = 0$$

5. Bias of Linear Approximation using Matsui's Piling up Lemma Equation :

```
            # calculate the bias of the linear approximation using Piling-Up Lemma Eqn
def piling_up_lemma(bias_list):
    mul_bias = bias_list[0]
    n = len(bias_list)
    for i in range(1, n):
        mul_bias *= bias_list[i]
    return 2 ** (n - 1) * mul_bias

bias = piling_up_lemma([-0.25, -0.375, -0.375])
No_of_plain_cipher_pairs_required = int(1/(bias**2))

Bias of Linear Approximation: -0.140625
Number of plaintext-ciphertext pairs required >= 50

```

Code 1.1: Lemma

6. The key bits targeted in this round are the first six Most significant bits of the key K4.

1. A 4-bit number (0111) is taken as input to $S - box_2$

2. Active S-boxes are :

- $S - box_2$:

$$X_2 \oplus X_3 \oplus X_4 = Y_2 \oplus Y_4, Probability(p) = (8-4)/16 = 4/16 = 1/4 Bias = p - 1/2 = -0.25$$

- $S - box_3$:

$$X_4 = Y_2 \oplus Y_3 \oplus Y_4 Probability(p) = (8+6)/16 = 14/16 = 7/8 Bias = ((p) - 1/2) = 3/8 = 0.375$$

- $S - box_4$:

$$X_4 = Y_2 \oplus Y_3 \oplus Y_4, Probability = 14/16 = 7/8 Bias = 7/8 - 1/2 = 0.375$$

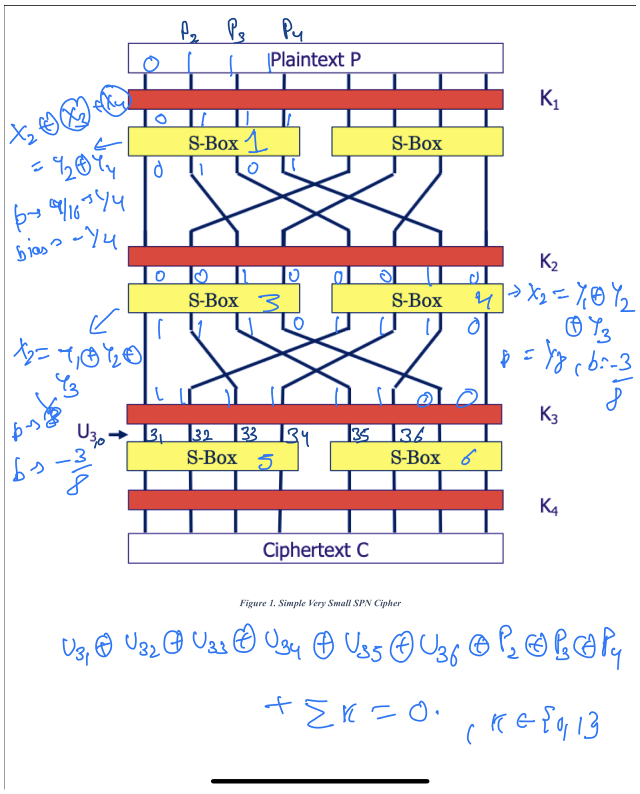

P_2 P_3 P_4
Plaintext P
K_1
S-Box 1
S-Box
K_2
S-Box 3
S-Box 4
K_3
U_{30}
S-Box 5
S-Box 6
K_4
Ciphertext C
Figure 1. Simple Very Small SPN Cipher
$X_2 \oplus X_3 \oplus X_4 = Y_2 \oplus Y_4$
$b \to 4/16 \to 1/4$
bias $\to -1/4$
$\to X_2 = Y_1 \oplus Y_2 \oplus Y_3$
$b = 1/8$, $b = -3/8$
$b \to -3/8$
$U_{31} \oplus U_{32} \oplus U_{33} \oplus U_{34} \oplus U_{35} \oplus U_{36} \oplus P_2 \oplus P_3 \oplus P_4 + \sum K = 0$, $K \in \{0,1\}$

Figure 1.3: *Approximation 1*

3. Combining the equations :

$$U_{33} \oplus U_{34} \oplus U_{35} \oplus U_{36} \oplus U_{37} \oplus U_{38} \oplus P_6 \oplus P_7 \oplus P_8 \oplus \sum K = 0$$

4. Bias of Linear Approximation using Matsui's Piling up Lemma Equation :

```
            # calculate the bias of the linear approximation using Piling-Up Lemma Eqn
def piling_up_lemma(bias_list):
    mul_bias = bias_list[0]
    n = len(bias_list)
    for i in range(1, n):
        mul_bias *= bias_list[i]
    return 2 ** (n - 1) * mul_bias

bias = piling_up_lemma([-0.25, 0.375, 0.375])
No_of_plain_cipher_pairs_required = int(1/(bias**2))

Bias of Linear Approximation: -0.140625
Number of plaintext-ciphertext pairs required >= 50

```

Code 1.2: Lemma

5. The key bits targeted in this round are the first six least significant bits of the key K4.

1.1 B

1.1.1 i. read-in the P-C pairs.

```
import math,random,string
import numpy as np
```

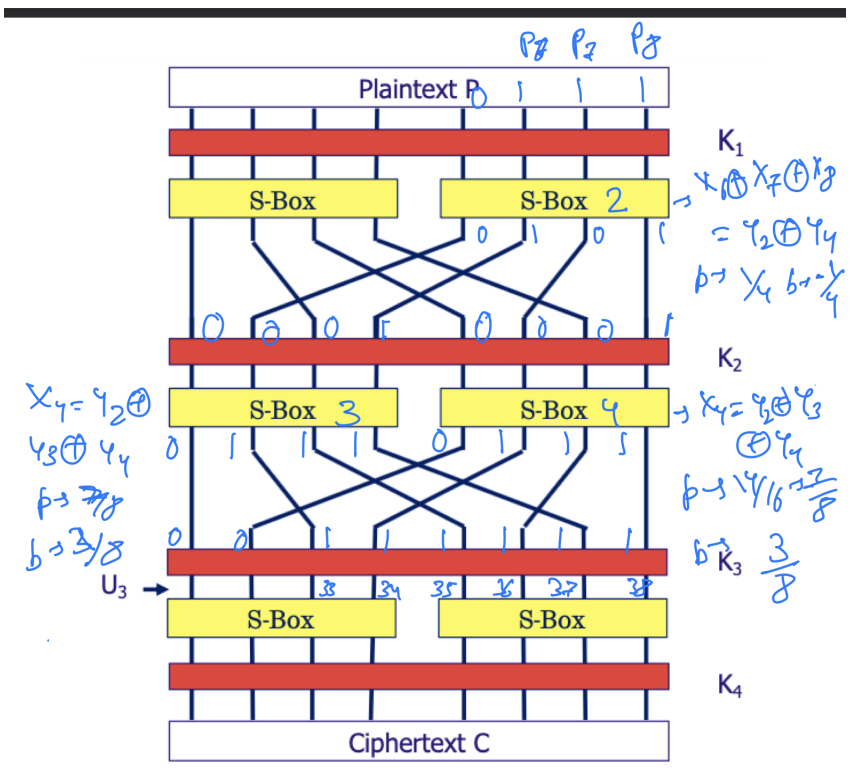


Figure 1. Simple Very Small SPN Cipher

$$U_{33} \oplus U_{34} \oplus U_{35} \oplus U_{36} \oplus U_{37} \oplus U_{38} \oplus P_5 \oplus P_7$$
$$\oplus P_8 \oplus \Sigma K = 0$$

Figure 1.4: *Approximation 2*

```python
from bitstring import BitArray
import pandas as pd
with open('D:\work\PC_pairs.txt', 'r') as data:
        x = []
        y = []
        for line in data:
            p = line.split()
            temp = np.binary_repr(int(p[0]), width=8)
            x.append(temp)
            temp = np.binary_repr(int(p[1]), width=8)
            y.append(temp)
plain = np.array(x)
cipher = np.array(y)
```

Code 1.3: P-C pairs

1.1.2 ii. carry out the (partial) decryption of ciphertexts (of PC-pairs) for a given (possibly partial) K4 key ‘guess’ (trial), i.e., to obtain the appropriate U3 bits, iii. carry out the linear approximation checking between appropriate P bits and U3 bits.

```python
input_size = 8
key_size = 2**8
s_box_input_size = 4
```

```python
s_box_config = {0x0:0xE, 0x1:0x4, 0x2:0xD, 0x3:0x1, 0x4:0x2, 0x5:0xF, 0x6:0xB, 0x7:0x8, 0x8:0x3, 0x9:0xA, 0xA:0x6, 0xB:0xC,
                0xC:0x5, 0xD:0x9, 0xE:0x0, 0xF:7}
s_box_config_rev = {0xE:0x0, 0x4:0x1, 0xD:0x2, 0x1:0x3, 0x2:0x4, 0xF:0x5, 0xB:0x6, 0x8:0x7, 0x3:0x8, 0xA:0x9, 0x6:0xA, 0xC:0xB,
                    0x5:0xC, 0x9:0xD, 0x0:0xE, 0x7:0xF}

keys = []
for i in range(256):
    keys.append(np.binary_repr(i,8))
def S_box(input,key,config):
        encrypted = 0
        #print('input',input)
        input_xored = np.binary_repr(np.bitwise_xor(int(input,2),int(key,2)),8)
        #print('xoree',input_xored)
        inputs_split_1 = np.binary_repr(int(input_xored[:4],2),4)
        #print(inputs_split_1)
        inputs_split_2 = np.binary_repr(int(input_xored[4:],2),4)
        #print(inputs_split_2)
        #print(s_box_config[inputs_split_1])
        boxxed_1 = np.binary_repr(config[int(inputs_split_1,2)],4)
        boxxed_2 = np.binary_repr(config[int(inputs_split_2,2)],4)
        #print(boxxed_1,boxxed_2)
        encrypted = boxxed_1 + boxxed_2
        return int(encrypted,2)
key_list = []
bias_list = []

for key in keys:
    count = 0
    for i in range(num_of_plain_cipher_pairs):
        plaintext = plain[i]
        ciphertext = cipher[i]
        u = S_box(ciphertext, key,s_box_config_rev)
        u31_34 = (u & 0xf0) >> 4
        u35_37 = (u & 0xf) >> 1
        u31 = (u31_34 & 0x8) >> 3
        u32 = (u31_34 & 0x4) >> 2
        u33 = (u31_34 & 0x2) >> 1
        u35 = (u35_37 & 0x8) >> 3
        u36 = (u35_37 & 0x4) >> 2
        u34 = (u31_34 & 0x1)


        p2_4 = (int(plaintext,2) & 0xf0) >> 4
        p2 = (p2_4 & 0x4) >> 2
        p3 = (p2_4 & 0x2) >>1
        p4 = (p2_4 & 0x1)

        xor_all = u31 ^ u32 ^ u33 ^ u35 ^ u36 ^ u34 ^ p2 ^ p3 ^ p4
        if (xor_all == 0):
            count += 1
    bias = abs(count / num_of_plain_cipher_pairs - 0.5)
    key_list.append(key)
    bias_list.append(bias)

result_df = pd.DataFrame.from_dict({'key': key_list, 'bias': bias_list})
result_df

row = result_df['bias'].argmax()
result_df['bias'][row]
row,result_df['bias'][103]


```

```python
""" SECOND APRROXIMATION : """
key_list = []
bias_list = []

for key in keys:
    count = 0
    for i in range(num_of_plain_cipher_pairs):
        plaintext = plain[i]
        ciphertext = cipher[i]
        u = S_box(ciphertext, key,s_box_config_rev)
        u31_34 = (u & 0xf0) >> 4
        u35_38 = (u & 0xf)
        u34 = (u31_34 & 0x1)
        u38 = (u35_38 & 0x1)
        u33 = (u31_34 & 0x2) >> 1
        u35 = (u35_38 & 0x8) >> 3
        u36 = (u35_38 & 0x4) >> 2
        u37 = (u35_38 & 0x2) >> 1


        p5_8 = (int(plaintext,2) & 0xf)
        p6 = (p5_8 & 0x4) >> 2
        p7 = (p5_8 & 0x2) >>1
        p8 = (p5_8 & 0x1)

        xor_all = u34 ^ u38 ^ u33 ^ u35 ^u36 ^ u37 ^ p6 ^ p7 ^ p8
        if (xor_all == 0):
            count += 1
    bias = abs(count / num_of_plain_cipher_pairs - 0.5)
    key_list.append(key)
    bias_list.append(bias)

result_df_2 = pd.DataFrame.from_dict({'key': key_list, 'bias': bias_list})
result_df_2
```

Code 1.4: U3 obtain

1.1.3 iv. identify promising candidates for K4. You should state in your answer why identified candidate(s) are particularly promising.

- From the First approximation, keys are '01101111' and'01100111' with both having maximum bias of 0.101562.
- For the second approximation, keys are '11100110' and '01100110' with a max bias of 0.15625.
- The most likely candidate for K4 is '01101110' or 102 in integer. as by flipping the last bit we get 01101111 which was approximated in the first approximation.

1.2 C.c) Outline how you would recover all remaining keys, i.e. K1 , K2 , and K3. (You are NOT expected to actually recover them, just outline how you would do this

- Exploit other linear approximations by running other approximations to get all final K4 bits, and reverse the last cipher-operations till second-last key-mixing
- Treat the cipher as having R-1 rounds and attack can be continued using Linear relations over R-2 rounds.
- Recursively solve till are keys are cracked.

1.3 d) How might small changes to the algorithm shown in figure 1 improve security?

The SPN cipher can be improved as follows:

1. The S-box suffer from a major weakness that they exhibit linearity, and as a result bias. To find a S-box combinations that reduce these two factors can make the cipher more secure.

2. The size of S-boxes can be increased as 'Quantity has a quality of its own' and can hide or obfuscate linear relationships more effectively.

3. as to attack this 3 round SPN cipher, we had to perform 2 round approximations, it would become harder to crack the cipher by Linear Crypt-analysis if the number of rounds are increased.

4. Using different S-box configurations akin to the DES cipher can also enhance security as it will make necessary approximation tables necessary for N-1 approximations.

5. Increasing input size and key size from 8-bits to 64bits.

Chapter 2

Question 2

2.1 a) Explain what is meant by the term ‘side channel’ in the context of crypto-systems.

[1] In the context of crypto-systems, a side channel is a method of attacking a crypto-system that does not involve trying to directly break the crypto-graphic algorithm or directly trying to guess the secret key. Instead, a side channel attack takes advantage of information leaked by the implementation of the crypto-graphic system, such as the time taken to perform a computation, the amount of power consumed, or the electromagnetic emissions produced.

For example, consider a device that uses a crypto-graphic algorithm to encrypt and decipher messages. An attacker might try to use a side channel attack by measuring the power consumption of the device while it is encrypting or deciphering a message. The attacker could then try to use this information to infer the key being used by the device, or to otherwise break the crypto-system.

Side channel attacks can be difficult to defend against, because they often do not rely on any weakness in the crypto-graphic algorithm itself. Instead, they exploit the way in which the crypto-graphic system is implemented or used. As a result, it is important for designers of crypto-graphic systems to be aware of the potential for side channel attacks and to take steps to minimize the amount of information that is leaked through side channels.

2.2 b) Give brief descriptions of particular side channels.

[2] Side channels can take many different forms, depending on the specific system being attacked and the information that is being leaked. Some common types of side channels include:

1. Timing attacks: These attacks exploit variations in the time it takes for a system to perform certain operations, such as encryption or decryption, to learn something about the internal state of the system. Researchers[2] int the paper " Remote timing attacks are practical " exploited this in their paper by sending carefully constructed cipher-text to the server and measured time it took to reject it when it encountered wrong padding, as the RSA cipher used Optimal Asymmetric Encryption Padding to make the cipher (padding + cipher + random) guard against multiplicative and deterministic weakness. The authors designed the cipher in such a way that it requires a subtraction operation when converting to a Montgomery representation. They also give three possible solutions to guard against this attack:

    - performing RSA blinding: "calculates $x = r^e g \mod N$ before decryption, where r is random, e is the RSA encryption exponent, and g is the cipher-text to be deciphered. x is then decipher as normal, followed by division by r, i.e.$x\frac{e}{r} \mod N$. Since r is random, x is random and timing the decryption should not reveal information about the key"
    - To make RSA decryption independent of cipher-text.
    - Carry out the dummy subtraction always, but not use it if not needed to guard against timing attack.

Other[3] measures to guard against timing analysis include making sure all conditional branches take same time to execute, adding noise to introduce random delays in program execution.

2. Power analysis[3]: These attacks measure the power consumption of a device as it performs crypto-graphic operations, in order to learn something about the internal state of the device.Simple Power Analysis and Differential Power Analysis are powerful tools used for implementing such attacks, To guard against them , the underlying principle remains same, that is conditional branches take comparable amount of power, to induce noise to minimise differences in power consumption based on input.

3. Electromagnetic analysis: These attacks measure the electromagnetic emissions of a device as it performs crypto-graphic operations, in order to learn something about the internal state of the device.

4. Acoustic crypt-analysis: These attacks measure the sound emitted by a device as it performs crypto-graphic operations, in order to learn something about the internal state of the device.Shamir[4] recovered an RSA key from a laptop using a microphone by using Acoustic crypt-analysis, they argue for solutions such as acoustic shielding, blinding and cipher-text normalisation.

5. Thermal Analysis: Temperature can also be used as a side channel, such as in a thermal imaging attack. This involves measuring infrared emissions from a processor during the execution of a cryptographic algorithm in order to infer some or all of the key bits

6. Fault injection : By varying the voltage or clock speed of the system, information can be leaked out, this has been used by Mordechai guri[5] in breaching Air-gapped computers and getting information out through covert channels.

7. Side channels can be a serious threat to the security of crypto-systems, and it is important to design systems that are resistant to these types of attacks. Cost-Benefit and threat perception will allow to make appropriate choices in deciding level of defensive measures..

Bibliography

[1] Tadayoshi Kohno, Niels Ferguson, and Bruce Schneier. *Cryptography engineering : design principles and practical applications*. Wiley Pub., inc., Indianapolis, IN, 2010. ISBN 9780470474242.

[2] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48:701–716, 01 2003. doi: 10.1016/j.comnet.2005.01.010.

[3] James Fell. Side channel attack countermeasures in cryptographic systems. *Github*, 1:1–3, 09 2015. doi: 10.1016/j.comnet.2005.01.010.

[4] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. Cryptology ePrint Archive, Paper 2013/857, 2013. URL `https://eprint.iacr.org/2013/857`. `https://eprint.iacr.org/2013/857`.

[5] Andy Greenberg. This Researcher Steals Data With Noise and Light. *Wired*. ISSN 1059-1028. URL `https://www.wired.com/story/air-gap-researcher-mordechai-guri/`.