



Hardware-Secured System for Secure Communications and Message Exchange

Alexandre Valente Rodrigues

Thesis to obtain the Master of Science Degree in
Information Systems and Computer Engineering

Supervisor: Prof. Ricardo Chaves

Month 2021

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetur erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis.

Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus.

Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.

I would also like to acknowledge my dissertation supervisors Prof. Some Name and Prof. Some Other Name for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

Individuals with high responsibility jobs such as government officials, top level company executives and diplomats are high profile targets to digital attacks, since they manage very sensitive information. Thus, attacks can have very damaging consequences for them and organizations. To maximize security, it is in their best interest to avoid storing cryptographic keys, passwords and perform critical cryptographic operations in their personal computers. This thesis proposes a cheap, relatively efficient but highly secure physical personal system, in a client-server mode, which enables individuals to securely exchange messages and sensitive documents. The proposed system secures communication by providing confidentiality and authentication to messages. This system will be responsible for performing every cryptography operation, store and manage cryptographic keys. All operations are performed inside the device and keys are never exposed to the outside, in order to not jeopardize the security of the communications.

Keywords

Communication Security; Secure Physical Device; Confidentiality; Authentication.

Resumo

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis. Aliquam aliquet, est a ullamcorper condimentum, tellus nulla fringilla elit, a iaculis nulla turpis sed wisi. Fusce volutpat. Etiam sodales ante id nunc. Proin ornare dignissim lacus. Nunc porttitor nunc a sem. Sed sollicitudin velit eu magna. Aliquam erat volutpat. Vivamus ornare est non wisi. Proin vel quam. Vivamus egestas. Nunc tempor diam vehicula mauris. Nullam sapien eros, facilisis vel, eleifend non, auctor dapibus, pede.

Palavras Chave

Colaborativo; Codificação; Conteúdo Multimídia; Comunicação;

Contents

1	Introduction	1
1.1	Problem	3
1.2	Requirements	3
1.3	Document Structure	4
2	Problem Definition	5
2.0.1	Context	7
2.0.2	Entities	7
2.1	Requirements	7
2.1.1	Goals	7
2.2	Services	7
3	Background	9
4	Related Work	11
5	System Architecture	13
5.1	Components	15
5.2	Operations	16
5.2.1	Administration Operations	16
5.2.2	Data Exchange Operations	16
5.2.3	Key Exchange Operations	17
6	Implementation	19
6.1	Initial State	21
6.2	Protocol	21
6.2.1	Authentication Protocol	21
6.2.2	Administration Protocol	22
6.2.3	Data Exchange Protocol	23
6.2.4	Key Exchange Protocol	25
A	Code of Project	29

List of Figures

5.1	Client and device	15
5.2	Client application and secure device	16
6.1	Authentication Protocol	22
6.2	Change Authentication PIN protocol	22
6.3	Data Exchange Encryption Protocol	23
6.4	Digital Signature Generation	24
6.5	Digital Signature Verification	25
6.6	Import Public Key	25
6.7	Protocol to generate new key to share with user.	26
6.8	Protocol to save key, received from another user.	26

List of Tables

- B.1 Example table 38
- B.2 Example of a very long table spreading in several pages 38

List of Algorithms

Listagens

A.1	Example of a XML file.	29
A.2	Assembler Main Code.	30
A.3	Matlab Function	31
A.4	function.m	32
A.5	HTML with CSS Code	32
A.6	HTML CSS Javascript Code	34
A.7	PYTHON Code	35

Acronyms

1

Introduction

Contents

1.1 Problem	3
1.2 Requirements	3
1.3 Document Structure	4

In the modern world, most people have access to a computer, involved in many everyday tasks, such as, web browsing, communications, social networks, news, entertainment, among many others. There is no limit to what you can achieve with the Internet, using just a computer. For this reason, computers have a wide range of attacks potentially exploitable by hackers, by taking advantage of software vulnerabilities or user mistakes. This is of great concern to people with high responsibilities from their jobs, who deal with sensitive information, such as, government officials, top level company executives and diplomats. Suffering an attack to a personal computer can be highly damaging as it can carry severe consequences for companies and countries. In addition, high profile officials who deal with sensitive information are more likely to be targeted by attackers.

1.1 Problem

New attacks, targeting computers, are discovered daily. They can come from zero-day vulnerabilities, phishing scams and many others, the opportunities are endless. It is impossible to predict and protect against all. Communications security, depends on the cryptography keys and passwords used. These are usually stored, along with other sensitive information, in the user's computer. Instead of storing the data in the computer, a more optimal solution, meaning, harder to compromise the security of communications, is to separate the platform used by the user for communications (their computer), and the device responsible for managing, securing communications and storing sensitive data. The goal is to add another layer of security, to make it difficult to compromise security even if the user's computer is compromised. A secure and independent solution is needed to establish secure channels of communication, store keys and perform critical operations, even if the computer might be compromised. A possible approach is the utilization of a personal physical device that is responsible for storing digital keys and perform critical operations. These devices need to be highly secure and independent from the user's personal computer.

1.2 Requirements

In order to address the problem and using the discussed approach, the implemented solution will have several requirements, to allow secure communications between multiple entities. It should perform all critical operations to the security of the communications, as well as, store all relevant secrets to the security of the interactions. A design requirement of the system is it should be easily usable to the regular user, with no technological expertise. The system must be efficient and low-cost, as so it is more easily accessible and scalable by interested users.

1.3 Document Structure

This first chapter introduces the context, the problem and basic requirements of the system. The second chapter goes into detail about the problem, its entities, devices, full requirements, goals of the system and the services it must provide. The third chapter will cover the technical background needed to comprehend the solution and state of the art. The fourth chapter will give context on the related work and existing solutions to the presented problem. The fifth chapter introduces the solution and its architecture. The sixth chapter will describe the system protocol and implementation.

2

Problem Definition

Contents

2.1 Requirements	7
2.2 Services	7

2.0.1 Context

2.0.2 Entities

These type of devices are aimed at people that handle very sensitive information, which have dire consequences if they are lost, corrupted or leaked. Some examples are government officials who handle confidential information pertaining to a country, company executives, such as the CEO who have access to company secrets, diplomats who manage confidential treaties, and military officers who have access to information critical to a countries' security.

2.1 Requirements

In order to address the problem and using the discussed approach, the implemented solution must comply with the following requirements:

- The box must be tamper-resistant, secure and personal;
- The system must allow secure interaction between different entities;
- All the user's secrets, such as keys and passwords must be stored in the device;
- The user's secrets must never be exposed to the outside;
- All critical operations must be performed in the device;
- The system must authenticate the user before performing operations. The system does not need to authenticate itself to the user;
- The system must be easy to use to the regular non-savvy user;
- The system should perform the operations in a reasonable time to minimize the user's wait;
- It should be relatively low cost.

2.1.1 Goals

2.2 Services

The solution will provide several services, which fulfill the requirements defined in 1.2.

- Secure Storage to save the user's sensitive cryptographic keys, passwords, documents or any data;

- Key management, generation, revocation and importation of keys when the users deem necessary;
- Confidentiality to keep the contents of the communications secret, except from authorized entities;
- Integrity to safeguard communications from unauthorized modifications;
- Authentication to ascertain the identity of the data sender;
- Non-repudiation to prevent an entity from denying authorship of a piece of information.

3

Background

4

Related Work

5

System Architecture

Contents

5.1 Components	15
5.2 Operations	16

The objective of the system was to develop a device in a box format to enable users to establish safe channels of communication. This is achieved with a safe and secure device which is personal to each individual. In order to secure the communications between users, the device saves the user's sensitive data, such as keys, and performs all security critical operations. The system is designed so that each user has it's own physical box.

5.1 Components

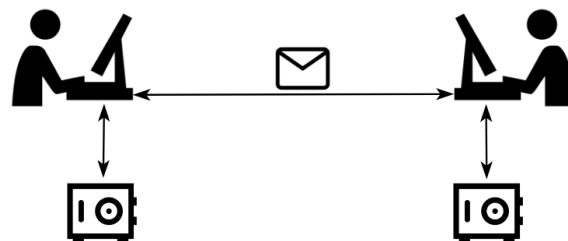


Figure 5.1: Client and device

The solution is composed of two main components, as shown in figure 5.1:

- The physical box which responsible for securing communications;
- The client application on the user's computer which provides an interface for the user to execute operations on the box.

By separating these components, the security of the system is isolated and solely of total responsibility of the box. It is not dependent on the user's personal computer.

Both components are connected through a common interface, such as USB, in order to be more easily accessible to the end users.

Figure 5.2 depicts the client application, interacting with the secure device through the application programming interface (API), the implementation of operations inside the device and secure storage where all the keys are stored.

Figure 5.2 depicts the client application, interacting with the secure device through the application programming interface (API), the implementation of operations inside the device and secure storage where all the keys are stored.

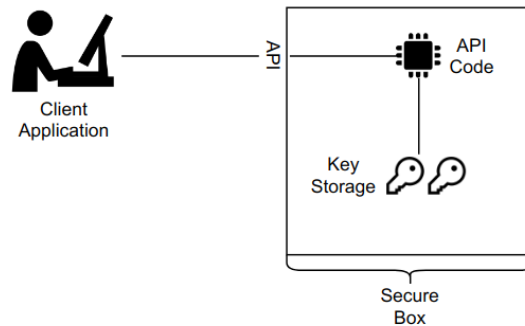


Figure 5.2: Client application and secure device

5.2 Operations

The system operations will ensure the system requirements and services are fulfilled. For the user to be able to execute them, he first must authenticate himself to the device. This is done with a PIN or password, which identifies the user. Once authenticated, the operations will be available to the user to be executed in the box.

The operations are split in three types:

- The administration operations manage the authentication and communication configuration;
- The data exchange operations secure the user's communication;
- The key exchange operations manage the keys stored inside the device, which will be used to secure communications.

5.2.1 Administration Operations

The administration operations will allow the user to manage the authentication related parameters. The only operations of this type is to change the authentication PIN. The device will be initialized from fabric with a default PIN which must be supplied to the user. Before performing any operation the user should change his PIN to begin secure communications.

5.2.2 Data Exchange Operations

The main operations will be responsible to secure the communications between users. These operations will fulfill the confidentiality, authentication and non-repudiation services.

- Secure data exchange with confidentiality and authentication. The objective of this operation is to send and receive data to and from the device. Plaintext data will be returned to the user encrypted

and authenticated with their key stored inside the device. In the case of encrypted and authenticated messages, an error will be returned if the decryption was unsuccessful, otherwise, the user will receive the plaintext data;

- Digital Signature operation will provide non-repudiation to a piece of data. The user will send the information to the box, and the subsequent signature will be returned, which can be used to verify the data's authorship. To verify a signature, the user sends it to the device, and receives either success or failure to verify.

5.2.3 Key Exchange Operations

These operations will handle key exchange when new keys need to be generated and exchanged between users, to enable further communications, and to import other user's public keys. This will serve the secure storage and key management services.

The first operations will enable the user to ask for a new key, generated inside the box, in order to securely send it to another user. The user receiving the new key, generated by another user, will receive and store the key inside the box. The final operation will provide a way to import other user's public keys, as well as export their personal public key, to be shared with another user.

6

Implementation

Contents

6.1 Initial State	21
6.2 Protocol	21

Symmetric keys will be used to encrypt and authenticate messages. They have better performance with larger messages compared to asymmetric keys. Each user can have stored in their box, several symmetric keys. This enables the user to establish secure communications with multiple different people or groups. The non-repudiation property of asymmetric keys will be used to create digital signatures of documents. The other use, will be to share symmetric keys between user who wish to communicate. The box stores the private and public key pair of the user, and the public keys of people the user wishes to trade secrets with.

6.1 Initial State

The users will receive the device with a pair of private and public keys, generated inside the device from fabric. Each device will have the user's public keys, whom he wishes to communicate. The user can request whose public keys he wants, before the device is initialized in fabric. This allows the users to trade symmetric keys between them, which they can use to begin trading data securely. In addition, the device can also come with a symmetric key stored in each the user's device.

For each user or group a user wants to communicate with, he has a symmetric key stored in the device.

When a new user wants to establish secure communications with an existing user or a group, he must share his public key with the user, ideally physically to ensure there are no mistakes or attacks. After this they can securely share symmetric keys to enable efficient and secure messaging.

6.2 Protocol

This section will explain and define the communication protocols between both components in more detail. For each operation, it will describe the different phases, what data is traded and why.

6.2.1 Authentication Protocol

Before executing any operation the user must authenticate himself to the device. The protocol described is pictured in figure 6.1.

1. The first phase is initiated by the user by sending a message to check if the box is alive and connected to the computer.
2. The operation will move to the second phase when the user receives an affirmative response. He will then send the operation code, which indicates he wants to authenticate himself, and the authentication PIN. The device will respond with a status parameter indicating failure or success.

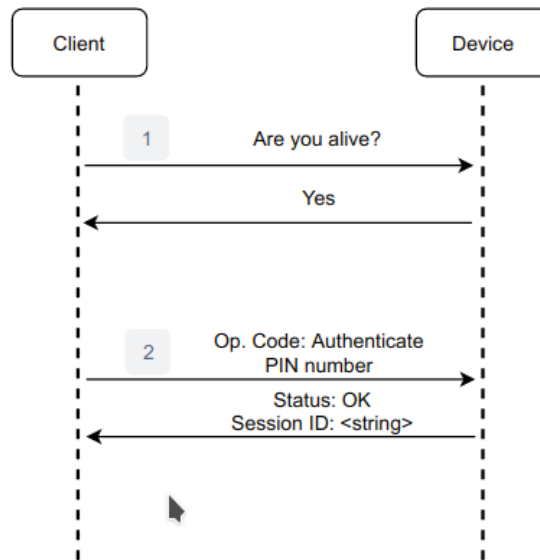


Figure 6.1: Authentication Protocol

When successful the box will also return a session ID string, which the user will need for further operations, to prove he has authenticated himself.

6.2.2 Administration Protocol

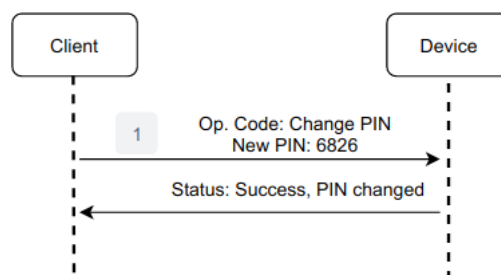


Figure 6.2: Change Authentication PIN protocol

As explained before, there is only one administration operation, changing the authentication PIN, pictured in figure 6.2.

The user initiates by sending the operation code, identifying the operation, the new PIN number and the session ID acquired previously. The device verifies the session ID and send a response, indicating the success or failure of the operation.

6.2.3 Data Exchange Protocol

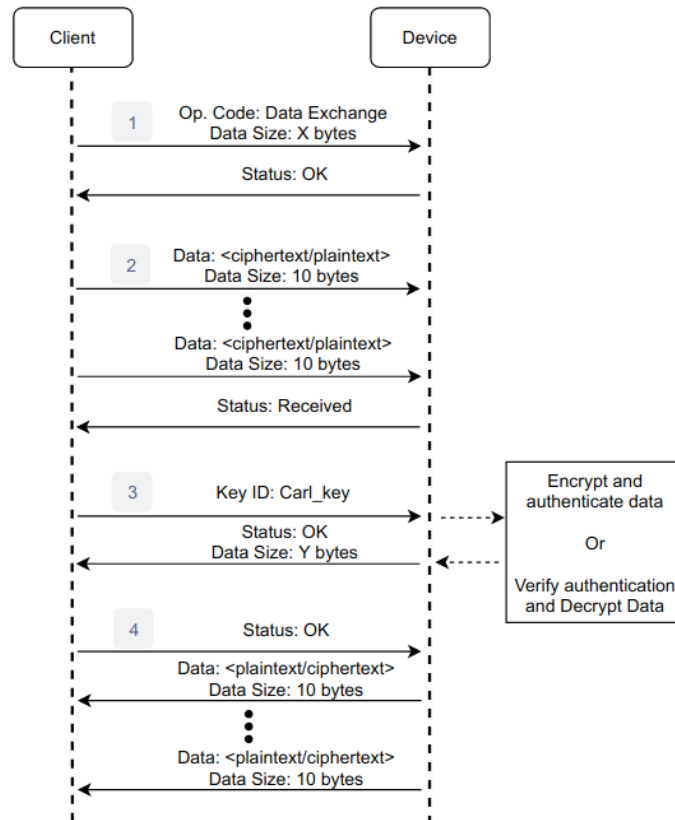


Figure 6.3: Data Exchange Encryption Protocol

The protocol to encrypt and authenticate data illustrated in figure ?? consists of:

1. The user sends the operation code and the data size, signaling he wants to send some data;
2. The box will respond with an OK message that the user can begin transmitting the data. It will be transmitted a maximum of X bytes per "packet". Each packet contains a part of the data and the size of the data in that packet. When the transmission ends, the device will confirm its reception;
3. The user subsequently will respond with the symmetric key ID, which he wants to encrypt and authenticate the data with. The box will handle the cryptographic operations and return a status message and the encrypted data size.
4. After the client confirms, the encrypted data with the additional MAC and IV parameters appended, will be returned in the same manner it was sent.

The protocol to decrypt and verify data authentication is very similar to the previous one, and is also pictured in figure 6.3.

1. The operation code is sent, as well as the encrypted data size;
2. The box will respond with an OK message that the user can begin transmitting the data, one packet at a time;
3. When the data transmission ends, the device will confirm its reception, and the user will subsequently respond with the symmetric key ID, which can decrypt and verify the data authentication;
4. After performing the decryption and authentication operations, the device will return a message indicating its success or failure. In case of a successful operations, it will return, in the same manner it was sent, the plaintext data.

In the case of digital signatures, the user's must have each others public keys, if they do not already have them.

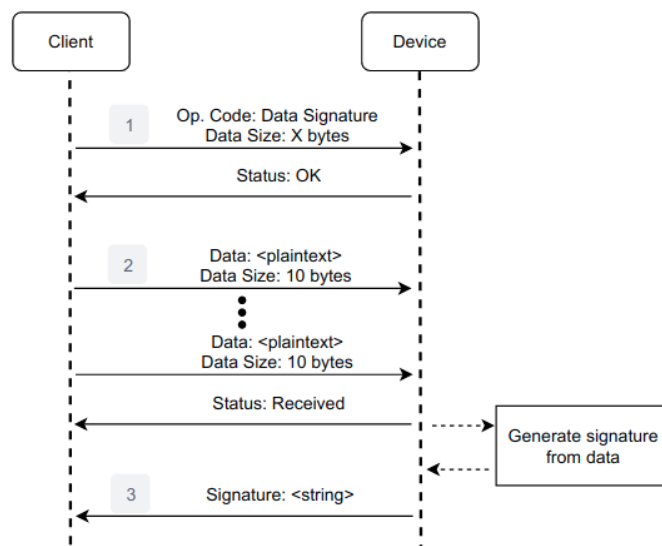


Figure 6.4: Digital Signature Generation

The next protocols are relating to the generation and verification of digital signatures. The designed protocol for generation is represented in figure 6.5.

The user initiates by sending the operation code and the plaintext data size. When the box responds with an OK message, the user transmits the data to be signed, one packet at a time. In possession of the data, the device will generate the digital signature using the user's private key. When finished the signature is sent back to the user.

The protocol for verifying digital signatures is pictured in figure 6.5.

After the user sends the operation code, and the box responds with an OK message, the user transmits the data, used by the signer to generate the signature, one packet at a time. When done,

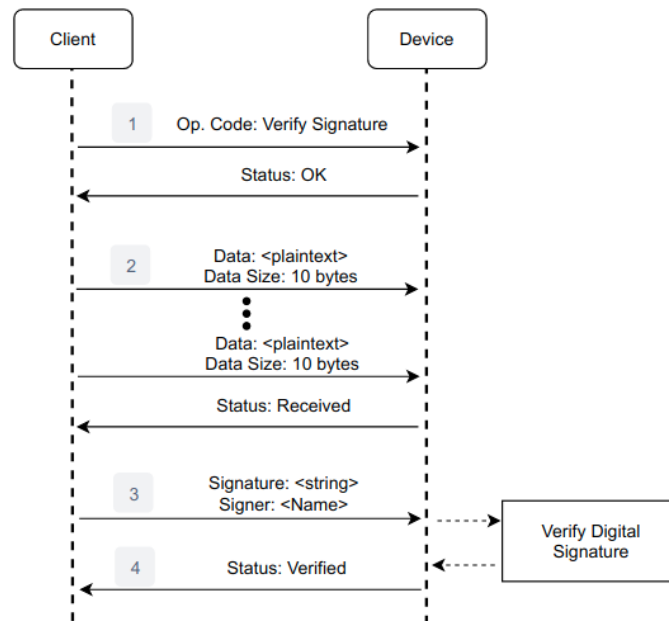


Figure 6.5: Digital Signature Verification

the user also sends the signature and the name of the signer, so the device knows what public key to use to verify the signature. Then, the device will verify the digital signature using the signer's public key, the data and the signature. The result will be sent back to the user.

6.2.4 Key Exchange Protocol

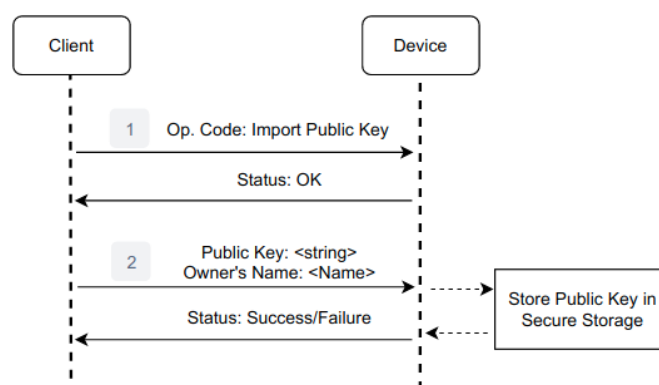


Figure 6.6: Import Public Key

Starting with the import public keys protocol, also represented in figure 6.6. The user send a message with the operation code, indicating he wants to store someone's public key. After the device responds

with an OK signal, the user sends the public key, and the name of the owner of the public key. The device, stores the public key, associated to the name sent by the user, and informs the user of the operation's success or failure.

Just like digital signatures, for users to be able to share symmetric keys between each other, they must possess each others public keys in their device. If not, they must physically meet to share them, and import them to their respective devices, with the available operation.

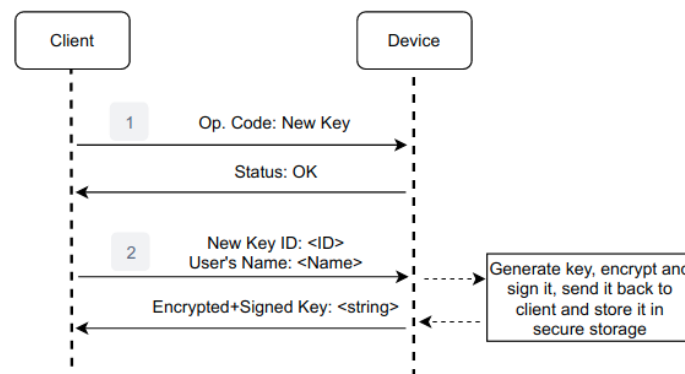


Figure 6.7: Protocol to generate new key to share with user.

The protocol to generate a new symmetric key, and securely share it with a user is represented in figure 6.7. The user sends a message with the operation code. After the device responds with an OK signal, the user sends the key ID, the name the key will be saved as, and the name of the user he wants to share the key with, so the device knows which public key to use to secure the key. A new symmetric key will be generated and saved in the device's secure storage, with the key ID sent by the user. The box will encrypt and sign the key with public-key cryptography, and send it to the user, which he can securely share with the other user.

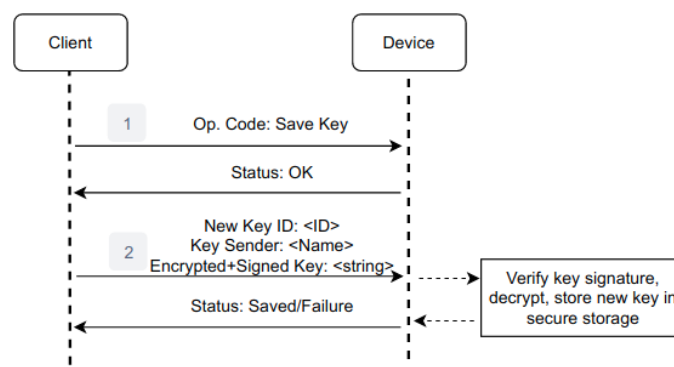


Figure 6.8: Protocol to save key, received from another user.

The protocol for the other user to save the newly received symmetric key, and store it inside their

device is in figure 6.8.

After the operations code is sent and the OK signal is returned, the user sends the key ID, the name of the key sender, and the encrypted and signed key. The device will then verify the signature with the sender's public key and decrypt the key, subsequently saving it in the device's secure storage along with other keys already present.



Code of Project

Nulla dui purus, eleifend vel, consequat non, dictum porta, nulla. Duis ante mi, laoreet ut, commodo eleifend, cursus nec, lorem. Aenean eu est. Etiam imperdiet turpis. Praesent nec augue. Curabitur ligula quam, rutrum id, tempor sed, consequat ac, dui. Vestibulum accumsan eros nec magna. Vestibulum vitae dui. Vestibulum nec ligula et lorem consequat ullamcorper.

Listagem A.1: Example of a XML file.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <StreamInfo version="2.0">
3   <Clip duration="PT01M0.00S">
4     <BaseURL>videos/</BaseURL>
5     <Description>svc_1</Description>
6     <Representation mimeType="video/SVC" codecs="svc" frameRate="30.00" bandwidth="401.90"
7       width="176" height="144" id="L0">
8       <BaseURL>svc_1</BaseURL>
9       <SegmentInfo from="0" to="11" duration="PT5.00S">
```

```

10         <BaseURL>svc_1-L0-</BaseURL>
11     </SegmentInfo>
12 </Representation>
13 <Representation mimeType="video/SVC" codecs="svc" frameRate="30.00" bandwidth="1322.60"
14     width="352" height="288" id="L1">
15     <BaseURL>svc_1/</BaseURL>
16     <SegmentInfo from="0" to="11" duration="PT5.00S">
17         <BaseURL>svc_1-L1-</BaseURL>
18     </SegmentInfo>
19 </Representation>
20 </Clip>
21 </StreamInfo>

```

Etiam imperdiet turpis. Praesent nec augue. Curabitur ligula quam, rutrum id, tempor sed, consequat ac, dui. Maecenas tincidunt velit quis orci. Sed in dui. Nullam ut mauris eu mi mollis luctus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Sed cursus cursus velit. Sed a massa. Duis dignissim euismod quam.

Listagem A.2: Assembler Main Code.

```

1  ; *****
2  ; * Constantes
3  ; *****
4
5  ON      EQU 1 ; contagem ligada
6  OFF     EQU 0 ; contagem desligada
7  INPUT   EQU 8000H ; endereço do porto de entrada
8          ;(bit 0 = RTC; bit 1 = botão)
9  OUTPUT  EQU 8000H ; endereço do porto de saída.
10
11
12 ; *****
13 ; * Stack
14 ; *****
15
16 PLACE   1000H
17 pilha:   TABLE 100H ; espaço reservado para a pilha
18 fim_pilha:
19
20 ; *****
21
22 PLACE   2000H
23
24 ; Tabela de vectores de interrupção
25
26 tab:     WORD    rot0
27
28 ; *****
29 ; * Programa Principal
30 ; *****
31
32 PLACE   0
33
34 inicio:
35     MOV BTE, tab ; incializa BTE
36     MOV R9, INPUT ; endereço do porto de entrada
37     MOV R10, OUTPUT ; endereço do porto de saída
38     MOV SP, fim_pilha
39     MOV R5, 1 ; inicializa estado do processo P1
40     MOV R6, 1 ; inicializa estado do processo P2
41     MOV R4, OFF ; inicializa controle de RTC
42     MOV R8, 0 ; inicializa contador
43     MOV R7, OFF ; inicialmente não permite contagem
44     EIO ; permite interrupções tipo 0

```



```

45 EI ; activa interrupções
46
47 ciclo:
48 CALL P1 ; invoca processo P1
49 CALL P2 ; invoca processo P2
50 JMP ciclo ; repete ciclo
51
52 ; *****
53 ;* ROTINAS
54 ; *****
55
56 P1:
57 CMP R5, 1 ; se estado = 1
58 JZ P1_1
59 CMP R5, 2 ; se estado = 2
60 JZ P1_2
61 sai_P1:
62 RET ; sai do processo.
63
64
65 P1_1:
66 MOVB R0, [R9] ; lê porto de entrada
67 BIT R0, 1
68 JZ sai_P1 ; se botão não carregado, sai do processo
69 MOV R7, ON ; permite contagem do display
70 MOV R5, 2 ; passa ao estado 2 do P1
71 JMP sai_P1
72
73 P1_2:
74 MOVB R0, [R9] ; lê porto de entrada
75 BIT R0, 1
76 JNZ sai_P1 ; se botão continua carregado, sai do processo
77 MOV R7, OFF ; caso contrário, desliga contagem do display
78 MOV R5, 1 ; passa ao estado 1 do P1
79 JMP sai_P1

```

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Phasellus eget nisl ut elit porta ullamcorper. Maecenas tincidunt velit quis orci. Sed in dui. Nullam ut mauris eu mi mollis luctus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos.

This inline MATLAB code `for i=1:3, disp('cool'); end;` uses the `\mcode{}` command.¹

Nullam ut mauris eu mi mollis luctus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Sed cursus cursus velit. Sed a massa. Duis dignissim euismod quam. Nullam euismod metus ut orci.

Listagem A.3: Matlab Function

```

1 for i = 1:3
2     if i >= 5 && a ~= b % literate programming replacement
3         disp('cool'); % comment with some  $\pi x^2$ 
4     end
5     [i,ind] = max(vec);
6     x_last = x(1,end) - 1;
7     v(end);
8     ylabel('Voltage ( $\mu V$ )');
9 end

```

¹MATLAB Works also in footnotes: `for i=1:3, disp('cool'); end;`

Nullam ut mauris eu mi mollis luctus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Sed cursus cursus velit. Sed a massa. Duis dignissim euismod quam. Nullam euismod metus ut orci.

Listagem A.4: function.m

```
1 % Copyright 2010 The MathWorks, Inc.
2 function ObjTrack(position)
3 % #codegen
4 % First, setup the figure
5 numPts = 300;           % Process and plot 300 samples
6 figure;hold;grid;       % Prepare plot window
7 % Main loop
8 for idx = 1: numPts
9     z = position(:,idx); % Get the input data
10    y = kalmanfilter(z);  % Call Kalman filter to estimate the position
11    plot_trajectory(z,y); % Plot the results
12 end
13 hold;
14 end % of the function
```

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Phasellus eget nisl ut elit porta ullamcorper. Maecenas tincidunt velit quis orci. Sed in dui. Nullam ut mauris eu mi mollis luctus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Sed cursus cursus velit. Sed a massa. Duis dignissim euismod quam. Nullam euismod metus ut orci. Vestibulum erat libero, scelerisque et, porttitor et, varius a, leo.

Listagem A.5: HTML with CSS Code

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Listings Style Test</title>
5     <meta charset="UTF-8">
6     <style>
7       /* CSS Test */
8       * {
9         padding: 0;
10        border: 0;
```

```

11     margin: 0;
12 }
13 </style>
14 <link rel="stylesheet" href="css/style.css" />
15 </head>
16 <header> hey </header>
17 <article> this is a article </article>
18 <body>
19     <!-- Paragraphs are fine -->
20     <div id="box">
21         <p>
22             Hello World
23         </p>
24         <p>Hello World</p>
25         <p id="test">Hello World</p>
26         <p></p>
27     </div>
28     <div>Test</div>
29     <!-- HTML script is not consistent -->
30     <script src="js/benchmark.js"></script>
31     <script>
32         function createSquare(x, y) {
33             // This is a comment.
34             var square = document.createElement('div');
35             square.style.width = square.style.height = '50px';
36             square.style.backgroundColor = 'blue';
37
38             /*
39              * This is another comment.
40              */
41             square.style.position = 'absolute';
42             square.style.left = x + 'px';
43             square.style.top = y + 'px';
44
45             var body = document.getElementsByTagName('body')[0];
46             body.appendChild(square);
47         };
48

```

```

49     // Please take a look at +=
50     window.addEventListener('mousedown', function(event) {
51         // German umlaut test: Berührungspunkt ermitteln
52         var x = event.touches[0].pageX;
53         var y = event.touches[0].pageY;
54         var lookAtThis += 1;
55     });
56     </script>
57 </body>
58 </html>

```

Nulla dui purus, eleifend vel, consequat non, dictum porta, nulla. Duis ante mi, laoreet ut, commodo eleifend, cursus nec, lorem. Aenean eu est. Etiam imperdiet turpis. Praesent nec augue. Curabitur ligula quam, rutrum id, tempor sed, consequat ac, dui. Vestibulum accumsan eros nec magna. Vestibulum vitae dui. Vestibulum nec ligula et lorem consequat ullamcorper.

Listagem A.6: HTML CSS Javascript Code

```

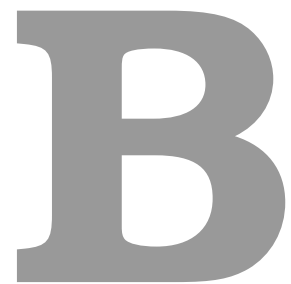
1
2 @media only screen and (min-width: 768px) and (max-width: 991px) {
3
4     #main {
5         width: 712px;
6         padding: 100px 28px 120px;
7     }
8
9     /* .mono {
10         font-size: 90%;
11     } */
12
13     .cssbtn a {
14         margin-top: 10px;
15         margin-bottom: 10px;
16         width: 60px;
17         height: 60px;
18         font-size: 28px;
19         line-height: 62px;
20     }

```

Nulla dui purus, eleifend vel, consequat non, dictum porta, nulla. Duis ante mi, laoreet ut, commodo eleifend, cursus nec, lorem. Aenean eu est. Etiam imperdiet turpis. Praesent nec augue. Curabitur ligula quam, rutrum id, tempor sed, consequat ac, dui. Vestibulum accumsan eros nec magna. Vestibulum vitae dui. Vestibulum nec ligula et lorem consequat ullamcorper.

Listagem A.7: PYTHON Code

```
1 class TelegramRequestHandler(object):
2     def handle(self):
3         addr = self.client_address[0]           # Client IP-address
4         telgram = self.request.recv(1024)       # Recieve telgram
5         print "From: %s, Received: %s" % (addr, telgram)
6         return
```

A Large Table

Aliquam et nisl vel ligula consectetur suscipit. Morbi euismod enim eget neque. Donec sagittis massa. Vestibulum quis augue sit amet ipsum laoreet pretium. Nulla facilisi. Duis tincidunt, felis et luctus placerat, ipsum libero vestibulum sem, vitae elementum wisi ipsum a metus. Nulla a enim sed dui hendrerit lobortis. Donec lacinia vulputate magna. Vivamus suscipit lectus at quam. In lectus est, viverra a, ultricies ut, pulvinar vitae, tellus. Donec et lectus et sem rutrum sodales. Morbi cursus. Aliquam a odio. Sed tortor velit, convallis eget, porta interdum, convallis sed, tortor. Phasellus ac libero a lorem auctor mattis. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nunc auctor bibendum eros. Maecenas porta accumsan mauris. Etiam enim enim, elementum sed, bibendum quis, rhoncus non, metus. Fusce neque dolor, adipiscing sed, consectetur et, lacinia sit amet, quam. Suspendisse wisi quam, consectetur in, blandit sed, suscipit eu, eros. Etiam ligula enim, tempor ut, blandit nec, mollis eu, lectus. Nam cursus. Vivamus iaculis. Aenean risus purus, pharetra in, blandit quis, gravida a, turpis. Donec nisl. Aenean eget mi. Fusce mattis est id diam. Phasellus faucibus interdum sapien. Duis quis nunc. Sed enim. Nunc auctor bibendum eros. Maecenas porta accumsan mauris. Etiam enim enim, elementum sed, bibendum quis, rhoncus non, metus. Fusce neque dolor, adipiscing sed, consectetur et, lacinia sit amet, quam.

Table B.1: Example table

Benchmark: ANN	#Layers (1)	#Nets (2)	#Nodes* (3) = 8 · (1) · (2)	Critical path (4) = 4 · (1)	Latency (T_{iter}) (5)
A1	3–1501	1	24–12008	12–6004	4
A2	501	1	4008	2004	2–2000
A3	10	2–1024	160–81920	40	60 [†]
A4	10	50	4000	40	80–1200
Benchmark: FFT	FFT size [‡] (1)	#Inputs (2) = 2 ⁽¹⁾	#Nodes* (3) = 10 · (1) · (2)	Critical path (4) = 4 · (1)	Latency (T_{iter}) (5)
F1	1–10	2–1024	20–102400	4–40	6–60 [†]
F2	5	32	1600	20	40 – 1500
Benchmark: Random networks	#Types (1)	#Nodes (2)	#Networks (3)	Critical path (4)	Latency (T_{iter}) (5)
R1	3	10–2000	500	variable	(4)
R2	3	50	500	variable	(4) × [1; ⋯ ; 20]

* Excluding constant nodes.

[†] Value kept proportional to the critical path: (5) = (4) * 1.5.

[‡] A size of x corresponds to a 2^x point FFT.

Values in bold indicate the parameter being varied.

As Table B.1 shows, the data can be inserted from a file, in the case of a somehow complex structure. Notice the Table footnotes.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetur. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

And now an example (Table B.2) of a table that extends to more than one page. Notice the repetition of the Caption (with indication that is continued) and of the Header, as well as the continuation text at the bottom.

Table B.2: Example of a very long table spreading in several pages

Time (s)	Triple chosen	Other feasible triples
0	(1, 11, 13725)	(1, 12, 10980), (1, 13, 8235), (2, 2, 0), (3, 1, 0)
2745	(1, 12, 10980)	(1, 13, 8235), (2, 2, 0), (2, 3, 0), (3, 1, 0)
5490	(1, 12, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
8235	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
10980	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
Continued on next page		

Table B.2 – continued from previous page

Time (s)	Triple chosen	Other feasible triples
13725	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
16470	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
19215	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
21960	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
24705	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
27450	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
30195	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
32940	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
35685	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
38430	(1, 13, 10980)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
41175	(1, 12, 13725)	(1, 13, 10980), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
43920	(1, 13, 10980)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
46665	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
49410	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
52155	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
54900	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
57645	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
60390	(1, 12, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
63135	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
65880	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
68625	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
71370	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
74115	(1, 12, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
76860	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
79605	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
82350	(1, 12, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
85095	(1, 12, 13725)	(1, 13, 10980), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
87840	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
90585	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
93330	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
96075	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
98820	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
101565	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
104310	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
107055	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
109800	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
112545	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
115290	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
118035	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
120780	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
123525	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
126270	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
129015	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
131760	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
134505	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
137250	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
139995	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
142740	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)
145485	(1, 12, 16470)	(1, 13, 13725), (2, 2, 2745), (2, 3, 0), (3, 1, 0)
148230	(2, 2, 2745)	(2, 3, 0), (3, 1, 0)

Continued on next page

Table B.2 – continued from previous page

Time (s)	Triple chosen	Other feasible triples
150975	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
153720	(1, 12, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
156465	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
159210	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
161955	(1, 13, 16470)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)
164700	(1, 13, 13725)	(2, 2, 2745), (2, 3, 0), (3, 1, 0)