TÉCNICO LISBOA

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 consectetuer 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; Codificaçãoo; Conteúdo Multimédia; Comunicação;

Contents

List of Figures

List of Tables

List of Algorithms

Listagens

Acronyms

1

Introduction

Contents

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

This chapter will define the problem, the client requirements and list the necessary functionalities of the solution, to successfully address the problem. First some context surrounding the problem is given. Next, the profile of the target clients will be described, and some examples will be given. The following section contains a list of client requirements the solution must abide by. Then it will shed the ligh on some essential concepts in order to understand the operations that need to be implemented. It will end by detailing those operations.

2.1 Context

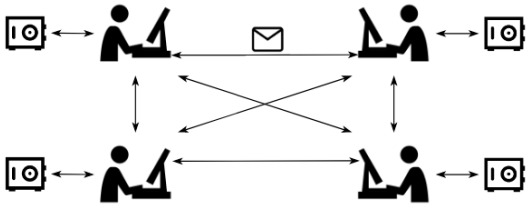

Figure 2.1: Client communication with their secure devices.

As discussed before, the same computers commonly used for communications and information storage are exploitable by attackers, and can cause a minor inconveniences, to possibly severe repercussions, such as, losing your confidential data to malicious parties.

An interesting approach to improve security is to add another layer of security to confidential data and communications through the addition of an device, independent of the user's personal computer. The device is responsible for the security of sensitive data and communications.

An illustration of the system with multiple users can be seen in figure 2.1.

2.1.1 Entities

These type of devices are especially relevant to people high responsibility jobs, 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.

Additionally, not just individuals have interest in these systems, a device can be assigned to a group of people representing an entity. For example, in the armed forces, a device can be assigned to the navy, one to the infantry, and every other faction. Any ranked officer, or people with a certain level of authority, could use the entity device, to communicate with other people or entities, in behalf of the group.

2.1.2 Devices

There are currently on the market some dedicated devices designed to secure communications and save private data. These type of devices have physical tamper-resistant measures against attackers who wish to read the device's information. They also provide fail-safe mechanisms in case of an attack. Hardware Security Modules (HSM) are high grade devices, with more computational power and larger storage capacity for the user's secrets.

Smart Cards, provide secure and portable tamper-resistant storage. They have lower processing power, and smaller memory which only allows to store a small amount of data. They have a low-cost, so can be produced in bulk and easily replaced. Only an RFID card reader is needed to read its information, and verify the owner's identity. Because of these features, they are widely used in the retail, healthcare, communication and government industries.

2.2 Client Requirements

To effectively address the presented problem, there are several high-level requirements the solution must adhere to:

- Devices should be distributable to either individuals, entities composed of multiple people or groups of people;
- The system must allow communications between groups of people and individuals representing themselves or an entity;
- The system must be responsible for securing all communications against disclosure, tampering or any sort of attacks;
- Users should be able to choose who they want to initiate secure communications. The application should provide the functionality to allow this;
- Is should be simple to use by everyone, including non-technical people;
- It should have a relatively low cost, enough to allow distribution of several devices between multiple individuals and groups;
- Only individuals with a certain level of clearance should be authorized to use the device. Personal devices should only be accessible by their owners.

2.3 Concepts

In this section some necessary concepts will be explained in order for non-technical users can easily understand the background, as well as the necessary concepts to fulfill the previously defined requirements.

Some services are crucial, in order to guarantee the security of communications.

Confidentiality is a security service which keeps the contents of communications secret, except from the authorized parties.

Integrity safeguards communications from modifications by attackers.

The **authentication** service can verify the identity of an agent, taking part in the communications.

Finally the **non-repudiation** service prevents an entity from denying authorship of a piece of information.

Cryptographic keys are tools used to grant the aforementioned services. Users in possession of the keys can secure and access their messages.

Symmetric keys, in possession of all communicating parties, are used to secure messages and documents.

Asymmetric key pairs (public and private key), are used to enable communicating by for example, sharing new symmetric keys between users who wish to communicate. Secondly, they provide non-repudiation through digital signatures. These keys identify the owner. The private key must always be in possession of the owner. With it they can prove their identity and generate signatures. The public key should be shared with other people so that they can share keys and verify the owner's signatures.

Digital signatures are a digital version of handwritten signatures, commonly used anywhere forgery detection is essential, for instance in financial transactions. Qualified signatures are a special type of signatures where the private keys are generated and stored inside a device, such as a Smart Card, and never leave it. This strong signature legally represents a person or a group. This type of signatures are used in the Portuguese Citizen Card.

2.4 Solution Services

This section will go into more technical detail on the services the solution should provide taking account the client requirements and the essential concepts.

2.4.1 communications

In order to secure communications, the following services must be guaranteed: confidentiality, integrity and authentication. The system must also give an option to provide non-repudiation to documents or

files, by means of digital signatures.

2.4.2 Key management

The device must store all the symmetric and asymmetric keys related to the entity or individual who owns the device.

The device must support secure storage in order to store the user's sensitive information, such as the cryptographic keys used for communication. Additionally, the device should have physical tamper-resistant measures and mechanisms in place, in case of an intrusion, such as, permanent erasure of all sensitive data. This means that even if an attacker is in possession of the physical device, it should be extremely difficult or even impossible to extract any information from it.

These keys must never be exposed to the outside environment of the device to ensure the security of communications and independence of the system.

All cryptographic operations must also be performed inside the device.

Key management operations should be supported, namely: symmetric key generation, symmetric key revocation, if communications are suspected to be compromised, and importation of other user's public keys.

Each entity has one pair of asymmetric keys, stored in their device, a private and a public. This pair identifies the entity. For each channel of communication between individual users, groups or entities, the same symmetric key is stored in both devices.

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, and establish a new secure communication channel when the keys are stored in their devices.

The users will receive the device with a pair of asymmetric keys, a private and public, 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 share symmetric keys between them, which they can user to begin trading data securely. The device can also come with the symmetric keys already shared and stored in each user device.

2.4.3 Authentication

Every user must authenticate himself, before using the device. This is done by providing a PIN code, which the device will verify before unlocking the session for the user.

The device will come from fabric with a default authentication PIN. This code can be changed by the users.

For personal devices, there is only the owner, but for groups and entities, there can be multiple users. In this case, there are two different ways to authenticate. The simplest is not to authenticate the person using the device, but have a single authentication PIN for the entity. All the user's with permission to communicate in behalf of the entity, must know the PIN code.

The second option, is to authenticate the user itself. The advantage of this approach is there can be a log of which users used the device and when, and what messages were sent and received for each user. This would entail a more complex process where, each user allowed to use the device, must register with a name and individual PIN code. The initial PIN code, would be used as an administration code, which allows registering users, and accessing the logs of user operations and message transactions.

It is worth nothing only the users are authenticated, the device does not authenticate itself to the user.

2.4.4 Usability

There are also several usability requirements the solution must abide by.

The solution should work with a plethora of devices, which will increase the adoptability of the solution among clients. This entails the use of a widely established protocol. **Public-Key Cryptography Standards (PKCS) #11** are a group of cryptographic standards, which define an application programmable interface (API) designed to interface between applications and cryptographic devices. By implementing the system in accordance with these standards, the solution will have device interoperability.

Another related requirement is the usage of a common connection solution, e.g. USB cable, to further increase the pool of supported devices.

The solution should provide an user facing interface which simple to use for the regular non-savvy user. In addition, the system should perform the operations in a reasonable time to minimize the user's wait, and improve the user experience.

3

Background

Contents

This section explains the necessary knowledge required to perfectly understand the problem, the proposed solution and the rationalization process behind it. It starts by providing an overview of cryptographic services, primitives and protocols. Then it presents several general purpose computing systems and ends by presenting other relevant components.

3.0.1 Cryptography

There are several cryptographic services relevant to this work, namely:

- Confidentiality: used to hide the content of information from unauthorized entities;
- Data Integrity: ability to detect the unauthorized modification of data;
- Authentication: used to ascertain the identify or origin of a message. Authentication is achieved if the combination of data integrity and freshness (prevention of replay attacks: when a valid message is replayed by a malicious entity);
- Non-Repudiation: prevents an entity from denying the authorship of a document or message.

To guarantee these services, there are two types of keys: symmetric and asymmetric. Symmetric keys are shared by 2 or more communicating parties. The keys are smaller and the operations are faster than with asymmetric keys.

Asymmetric keys constitute a pair for each party, one private and one public key. The private key is personal to its owner and should never be shared. The public key may be shared widely to other parties. Asymmetric keys are bigger and the operations are slower compared to symmetric key algorithms.

There are two types of ciphers regarding the procedure: stream and block ciphers. Stream ciphers generate an infinite stream of pseudo-random bits as the key, know as key-stream. The stream is used to encrypt, usually 1 bit of plaintext at a time. The operation to combine the key-stream and plaintext is an exclusive-or(XOR). Stream ciphers are usually faster than block ciphers, have lower memory requirements and are therefore cheaper and more suitable to embedded devices with limited memory. However they are prone to weaknesses based on usage, in particular, using the same initialization cector (IV) more than once.

Block ciphers encrypt fixed-length groups of bits, called blocks, with a symmetric key. They have a higher memory usage, in order to keep the blocks in memory. Since the plaintext is encrypted one block at a time, if the plaintext length is not a multiple of the block size, the last block needs to be padded. This can be taken advantage of by attackers if not done correctly. Another side effect of using blocks is, it will be more susceptible to noise in transmissions. If a bit is flipped with a stream cipher, only the corresponding bit is affected. While with a block cipher, more than 1 bit is affected, depending on the mode.

Symmetric ciphers support both block and stream ciphers while asymmetric use block ciphers.

3.0.1.A Hash Functions

A cryptography hash function generates a fixed dimension value (digest) based on variable input texts, such as messages or files. Secure hashes provide message integrity by comparing digests, calculated before, and after, transmission to determine if the message was altered. To achieve this, hash functions must have several properties:

1. They must be deterministic, meaning the same input value must always result in the same hash value;
2. They must generate very different output values for similar inputs;
3. They must be collision resistant, meaning it should be hard to find two input messages that generate the same hash value;
4. The hash value should be computed relatively quickly;
5. Given a hash value, it should be hard to find an input text that produces that hash value.

Popular hash functions include MD5, SHA-1, SHA-2 and SHA-3. Collisions against MD5 can be performed in seconds and have been produced against SHA-1 as well. Thus, these hash functions are considered broken and should not be used. Currently, SHA-2 and the newest SHA-3 are recommended. The SHA-2 algorithm provides different functions which vary on the size of the digest. For example, the SHA-256 function produces a 256 bit digest and SHA-512 a 512 bit digest. All functions are secure for the foreseeable future, there is no practical security advantage in using a bigger digest.

3.0.1.B Symmetric Encryption.

Symmetric ciphers use symmetric keys and are frequently used to achieve, data integrity, authentication and confidentiality.

One of the most popular symmetric-key algorithms is the Advanced Encryption Standard (AES). This algorithm uses 128-bit blocks for block cipher modes and the keys can be 128, 192 or 256 bits. AES has block and stream cipher modes.

Electronic codebook (ECB) is the simplest mode. It is a block cipher and works by encrypting each block with the symmetric key. If the same key is used, for equal plaintext blocks, the result will always be the same. For this reason, patterns are easily seen and the mode is considered insecure.

Cipher block chaining (CBC) is another block cipher mode. It combines the first block of plaintext and an IV with the XOR operator and encrypts the result. For the subsequent blocks, the previous ciphertext is used instead of the IV. The message needs to be padded to a multiple of the block size. If

this is not done correctly, it can be exploited with a padding oracle attack [**?**]. Implementing ciphertext stealing, resolves the issue and is recommended for the security of CBC [**?**]. Encryption is not parallelizable since a ciphertext block depends on all the blocks before it. Another disadvantage of CBC is it cannot precompute data to improve encryption performance. To decrypt a ciphertext block, only the previous ciphertext block is needed. Therefore random read access is supported and decryption can be parallelized. Regarding error propagation, when a bit is flipped in the ciphertext, the plaintext block will be completely corrupted and the corresponding bit of the next block will be inverted.

The **Output feedback (OFB)** mode repeatedly encrypts the IV for each block, xoring the result with the plaintext block. The encryption and decryption processes are exactly the same. The block cipher is only used in the encryption direction, which means the message does not need to be padded. It is effectively a stream cipher. The downside of needing to encrypt the IV multiple times, is the encryption and decryption are not parallelizable and random read access is not possible. However, the multiple encryptions of the IV can be precomputed in order to increase the performance of both encryption and decryption. Changes to a ciphertext block, only affect the corresponding bits.

Cipher feedback (CFB) is a combination of OFB and CBC. It xors the encrypted IV with the plaintext block. The result is then used for the next block. Similarly to OFB, CFB is effectively a stream cipher. Yet, akin to CBC, and for the same reasons, encryption is not parallelizable, opposed to decryption which is. Random read access is possible, contrary to preprocessing which is not. An interesting difference of CFB is it can become self-synchronous, meaning it will recover if s bits are lost. If s=1, it can recover from slips of any number of bits, if s=8 it can recover from slips of any number of bytes. However for every blockcipher call, CFB only processes s bits, compared to 128 bits (block size), which is a major performance cost.

Counter (CTR) mode concatenates an IV with a counter beginning at 0. Each sequence is encrypted and xor'ed with the plaintext block. For each block the sequence is incremented by 1. This mode is comparable to OFB, as it is also a stream cipher, the encryption operation is exactly the same as the decryption and an error affects only the respective bits. However it does not have the performance disadvantages of OFB. Encryption and decryption are parallelizable, random read access and preprocessing are both possible. It is worth noting that due to existing no need to implement decryption, it can save hardware costs and simplify code.

For every mode with an IV, it needs to be sent along with the message to the receiver, or the receiver will not be able to retrieve the entire message.

CBC, OFB and CFB modes are proved secure, assuming the IV is random, and is unique, meaning it is only used once for each key and message. For CTR mode, the IV does not need to be random, but it cannot be reused with the same key. After the IV is used, there is no need for the value to be kept secret. It can be sent alongside the ciphertext.

Regarding performance, CBC is slower than CTR mode. CFB (even with s = 128) and OFB are slower still. When efficiency characteristics matter, nothing comes close to CTR: it has better performance characteristics, in multiple dimensions, than any of CBC, CFB, and OFB.

All these cipher modes are malleable, meaning an attacker can modify a ciphertext C, the result of encrypting plaintext P, to create ciphertext C' which will decrypt to plaintext P' that is similar to P. Malleability is connected to message integrity. This is not considered a relevant weakness since the modes only goal is to offer confidentiality guarantees, not integrity. If one wishes integrity it should pair one of these modes with a MAC (Message authentication code), or use a dedicated authenticated-encryption mode like CCM or GCM (discussed in Section 3.0.1.D), which guarantee both confidentiality and integrity.

3.0.1.C Message Authentication Code (MAC).

MAC is a value, also called tag, used for authenticating a message. A MAC algorithm, receives the message and a symmetric key, to generate a tag. Unlike digital signatures, MAC do not offer non-repudiation since it uses a symmetric key, which need to be distributed to all parties. Any of the users in possession of the key can generate a MAC for a message, as well as verify it. On the contrary, digital signatures utilize the private key from asymmetric cryptography, which is personal.

Several techniques exist to construct a MAC. One is cipher block chaining message authentication code (**CBC-MAC**), which utilizes the CBC block cipher to encrypt data. A chain of blocks is generated, and the last block is the tag. CBC-MAC also has similar caveats to CBC, it is only secure for fixed-length messages [**?**] and different keys have to be used for CBC encryption and generating the authentication tag. CBC-MAC security deficiencies were resolved with One-key MAC (OMAC), which is secure for variable-length messages.

HMAC is different from the previous techniques, by using a cryptographic hash function, such as SHA-2, and a symmetric secret key to construct a tag. HMAC is secure, as long as the underlying hash function used is considered secure. Therefore SHA-2 is a good option. Despite CBC's inefficiencies discussed in section 3.0.1.B, specifically, each block is serially encrypted, HMAC is slower due to the inefficient hashing operations. However, HMAC does not have the security problems of CBC-MAC. HMAC is a popular and well-designed construction, but it is not the most efficient approach [**?**].

3.0.1.D Authenticated Encryption.

Authenticated encryption with associated data (AEAD) schemes assure both confidentiality and authenticity using only symmetric keys. They may be more efficient than combining separate privacy and authentication techniques, such as the ones discussed in earlier sections, and are less likely to be used incorrectly. AEAD schemes also allows associated data to be included in the message, which is authen-

ticated but not encrypted. This feature is useful, for example, for network packets. The header is visible but is authenticated. The payload is both authenticated and encrypted.

Counter with CBC-MAC mode (CCM) is an AEAD mode that combines CBC-MAC for authentication and CTR for encryption. CCM uses a MAC-then-Encrypt approach. First, CBC-MAC is computed on the message to obtain the tag. Then the message and the tag are encrypted with CTR mode. Due to performing two encryption operations, CBC-MAC and then CTR, it is a less efficient mode compared to others such as GCM, which only performs one encryption operation. It is not an online mode, meaning it needs to know the message and Associated Data (AD) length beforehand. Therefore, AD cannot be preprocessed. It is only considered secure for fixed-length messages. Despite being a slower mode, it is secure and achieves its goals, so it is widely supported, included in IPsec, TLS and Bluetooth low energy.

EAX mode aims to improve on CCM by replacing CBC-MAC with OMAC. Similarly to CCM, it first generates the authentication tag with OMAC, then encrypts with CTR. By using OMAC instead of CBC-MAC, it supports variable-length messages and is online.

Galois/Counter Mode (GCM) utilizes an encrypt-then-MAC approach. It first encrypts with CTR mode, then uses Galois mode of authentication to generate the tag. The Galois field multiplication supports parallel computation, making this mode faster than CCM. Evidently, like in normal CTR mode, it needs a different IV for each encrypted message. Beyond being parallelizable, it has the advantages EAX has over CCM. It is online and the AD can be preproccesed. For security reasons, authentication tags should be at least 96 bits, even though the mode allows smaller tags. One limitation of GCM is, it can encrypt a maximum of 64GiB of plaintext. Security analysis of several modes, decisively states that GCM in hardware is unsurpassed by any authenticated-encryption scheme.

3.0.1.E Asymmetric Encryption.

Asymmetric cryptography utilizes a pair of public and private keys. It is commonly used to provide confidentiality, data integrity, authentication and non-repudiation. The private keys must always remain secure with the owner. Public keys may be distributed as it does not compromise security. Encrypting a message with the public key, provides confidentiality, since only the owner who possesses the private key, can decipher the message. On the other hand, private key encryption provides authentication on account of only the owner is in possession of the private key. These two different concepts can be combined to provide confidentiality, authentication and non-repudiation to a message.

Compared to symmetric keys, asymmetric keys are less risky to distribute, as the public key can be viewed by anyone. However, there is the problem of validating public keys, which consists of guaranteeing a public key is owned by the correct identity.

Once two parties have traded public keys, asymmetric and symmetric keys can be combined in a

hybrid encryption scheme. The scheme takes advantage of the faster symmetric encryption to cipher the data, and the asymmetric encryption to encrypt the symmetric key, and provide authentication. Alternatively, it can be used to share symmetric keys for usage with an authenticated-encryption scheme.

There are two popular algorithms for public-key encryption, Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC). RSA has been used for decades, it is well established and widely used. It is based on the difficulty of factoring the product of two large prime numbers.

ECC is a more recent algorithm, based on the Elliptic Curve Discrete Logarithm Problem. For the same level of security, ECC keys are smaller. Gupta & Silakari, 2011 [**?**], give as an example a 160-bit ECC key has similar security to a 1024-bit RSA key. They also state that smaller key sizes may result in faster execution timings for the schemes, which is beneficial to systems where real time performance is a critical factor. With the threat of quantum computers, both ECC and RSA could become obsolete in the future, as it is vulnerable to brute force attacks from such devices.

3.0.1.F Digital Signatures.

Signatures is a standard scheme for authenticating documents or digital messages and ensuring the signer cannot repudiate the signature. The digital signature is generated by a combination of asymmetric keys and hash functions. The digital signature is generated by first computing a hash of the message, then signing the hash with the author's private key. The message is not directly signed since public-key encryption is slow and messages are most likely bigger than the hash of the message, which has a fixed size. Third parties can validate the signature with only the signature and the author's public key. Only the author is in possession of their private key, so only he could have generated the signature. They are a digital version of handwritten signatures [**?**], commonly used anywhere forgery detection is essential, for instance in financial transactions or software distribution.

Qualified signatures are a special type of signatures where the private keys are generated and stored inside a device, such as a Smart Card, and never leave it. For the owner to sign a document, he needs to be in possession of the Smart Card (something you have) and a PIN code (something you know). This strong signature legally represents a person or a group. This type of signatures are used in the Portuguese Citizen Card.

3.0.1.G Public Key Infrastructure

Asymmetric cryptographic needs a secure mechanism to validate public keys, i.e guaranteeing a public key is owned by a certain identity. A Public Key Infrastructure (PKI) is a central database of public-key certificates. It is responsible for managing, distributing, storing and revoking digital certificates. Digital certificates map public keys to identities and are used to verify that a specific public key belongs to a certain identity. A PKI has several components e.g. a registration authority, a certification authority and

a central database of stored keys. A user can also submit other entities' public keys. Other entities that trust the user responsible for the submission, can use the public keys to authenticate messages. There are alternative approaches to PKI, such as a web of trust. This mechanism self-signs certificates and third parties attest these certificates. This approach is implemented in PGP [**?**].

Transport Layer Security (TLS) is a cryptographic protocol that aims to provide confidentiality and data integrity, during transmission, over TCP/IP. It uses symmetric cryptography to encrypt data. A new symmetric key is generated for each connection. TLS supports asymmetric cryptography which authenticates the identity of the communicating parties. TLS is widely used in web browsing, e-mail and instant messaging. The protocol can provide perfect forward secrecy, unlike PGP, assuring any past connections are secure, if in the future encryptions keys are disclosed.

3.0.2 General Purpose Computing Systems

In this section we discuss some general purpose computing systems that may be pertinent to this work.

Secure cryptoprocessors are dedicated physical computational devices for performing cryptographic operations. Some secure cryptoprocessors namely, Smart Cards, Trusted Platform Modules and Hardware Security Modules will be discussed next.

3.0.2.A Hardware Security Modules (HSM).

A HSM is a high grade computational device responsible for storage, management and generation of cryptographic keys, as well as performing cryptographic operations. Keys never leave the device and all operations are performed inside the HSM. These devices have physical security mechanisms to achieve tamper-resistance, support several cryptographic operations, random number generators and have fail-safe mechanisms in place, in case of an attack, e.g., deletion of keys. Some devices have accelerated Central Processing Units (CPU) and optimizations to improve operations' performance. These modules are usually costlier than other computational systems i.e. Trusted Platform Modules, but are more advanced in processing power and available operations.

Smart Cards.

Smart Cards are a type of HSM, credit card-sized with an embedded microchip and provide secure, tamper-resistant storage. These devices have a low price for manufacturing, which allows for bulk production of the device and easy replacement if needed. However, the disadvantage is it makes it easy for attackers to acquire many devices to try to tamper with. They have a low processing power, and small memory which only allows to store a small amount of data. To be authenticated, the cards need readers

that are either contact or contactless (RFID technology). These characteristics make them extremely popular, used in many industries, such as, retail, healthcare, communication and government.

3.0.2.B FPGA System-on-Chip.

A Field-Programmable Gate Array (FPGA) is an integrated circuit designed to be programmed and configured after manufacturing. FPGAs are often used to prototype and for high specialized systems produced at low scale. One of the major advantages is their agility and flexibility to be customized for special use cases [**?**]. However, the reconfigurability may introduce certain weaknesses to the system. FPGAs are generic and its bitstream is vulnerable to cloning if no additional protection is applied. Cloning is simple and is considered the worst security vulnerability of volatile FPGAs [**?**]. The configuration data of these devices is stored in non-volatile memory and may be directly copied if no authentication mechanism is implemented [**?**].

An example of such device is the SmartFusion2 System-on-Chip from Microsemi that delivers more resources in low-density devices with the lowest power, proven security, and exceptional reliability [**?**]. Smartfusion2 SoC integrates a non-volatile FPGA with a system-on-chip (SoC) and an internal secure eNVM for storing Phase 0 boot code. Since they are non-volatile, the bitstream is at a lower risk of being probed during boot [**?**]. The eSRAM flash memory is protected against single event upsets. Smartfusion2 has an embedded ARM Cortex-M3 processor and a true random number generator (TRNG), which provides a quality source of entropy, a critical part of most cryptographic algorithms. It supports some cryptographic functions: AES-256, SHA-256, ECC and Physically Unclonable Function (PUF) (explained in Section 3.0.3.B).

Secure boot.

Not validating code before its execution leads to potentially executing untrusted code. This causes problems such as hackers inserting malware to hijack systems, download intellectual property, spy on users or perform any number of attacks. Most embedded processors do not validate code before it is executed. The SmartFusion2 SoC solves this problem by using a non-volatile flash memory (eNVM) to store boot code which can be write protected. Another reason is it authenticates each stage of the boot process to create a chain-of-trust. Other systems also implement solutions to secure boot code. Infineon secures the boot process for ARM platforms by incorporating a Trusted platform Module (TPM), compliant with version 1.2, into the processor. The TPM operates as a root of trust to certify the platform's integrity and correct system state. This prevents tampered kernels and fault attacks. Texas Instruments Sitara processors allows the customer to specify a public key as a root of trust to be fused into the device. This key is used to authenticate other keys that can be used to authenticate software components.

3.0.3 Other Important Concepts

There are some important concepts to consider involving cryptography and genera purpose computing systems.

3.0.3.A Random Number Generators

are components that generate a sequence of numbers that cannot be predicted. Generating random numbers is a common and critical requirement for almost all cryptographic algorithms. Pseudo-random number generators are frequent in software approaches and are not truly random. They depend on an algorithm and initial conditions (seed) to generate random numbers. If the seed is known, the numbers are predictable. True random number generators (TRNG) are hardware devices that generate numbers from microscopic physical conditions. These conditions are completely unpredictable. For this reason, TRNGs are perfect for use in cryptography. HSM are usually equipped with a TRNG.

3.0.3.B PUFs

are unique identifiers for microprocessors like HSM and smart cards. They depend on unpredictable physical factors of the device introduced during manufacturing, so are difficult to predict and replicate even on identical hardware. PUFs implement a challenge-response authentication system. A challenge value is sent to the device and an unpredictable response is computed by the PUF. Several pairs of challenge, response values are allowed. PUFs can also be used for key generation and as a source of randomness.

3.0.3.C Public-Key Cryptography Standards #11.

Public-Key Cryptography Standards (PKCS) are a group of cryptographic standards, published by RSA Laboratories, that describe guidelines for using cryptographic schemes. PKCS#11 defines an application programmable interface (API) designed to interface between applications and cryptographic devices such as smart cards and hardware security modules [**?**]. The standard has been widely used, promoting interoperability between devices. By using the same standard, it allows devices to take advantage of another device's API. The PKCS#11 API allows applications to access cryptographic devices through slots. The slots represent a socket or device reader. A session can be established through the slot so the application can authenticate itself to a token with a default PIN. The token holds private and public objects which can be keys and certificates. Only public objects are available to an unauthenticated session. The normal user has access to both private and public objects but cannot change the authentication PINs. The privileged user can change both his and the normal user's PIN. An attack scenario

is possible if the host machine is compromised, an attacker can intercept the PIN, which he can use to access the cryptographic device.

4

Related Work

5

System Architecture

Contents

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

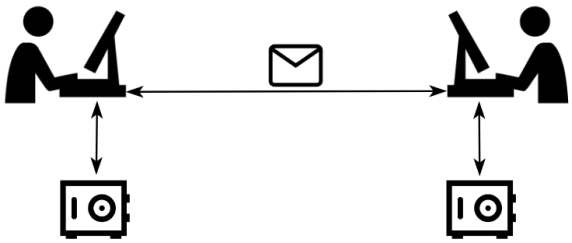

Figure 5.1: Client and device

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.

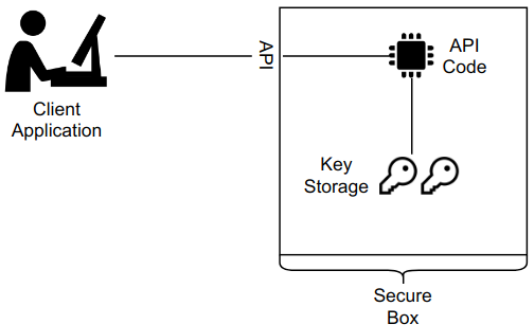

Figure 5.2: Client application and secure device

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

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 user 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.

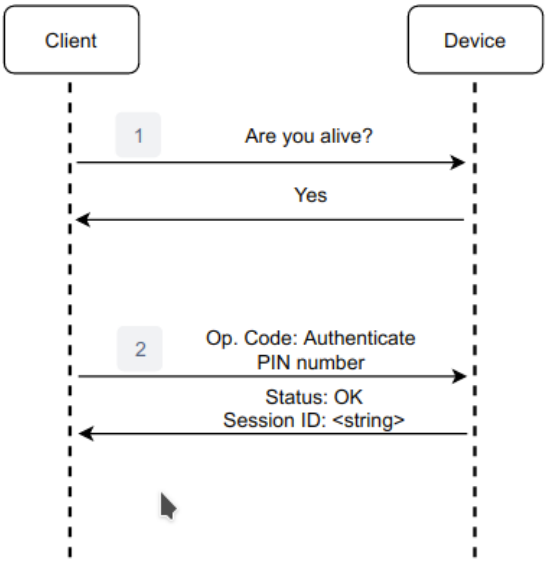

Figure 6.1: Authentication Protocol

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

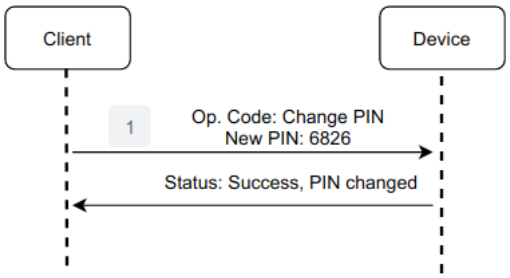

Figure 6.2: Change Authentication PIN protocol

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

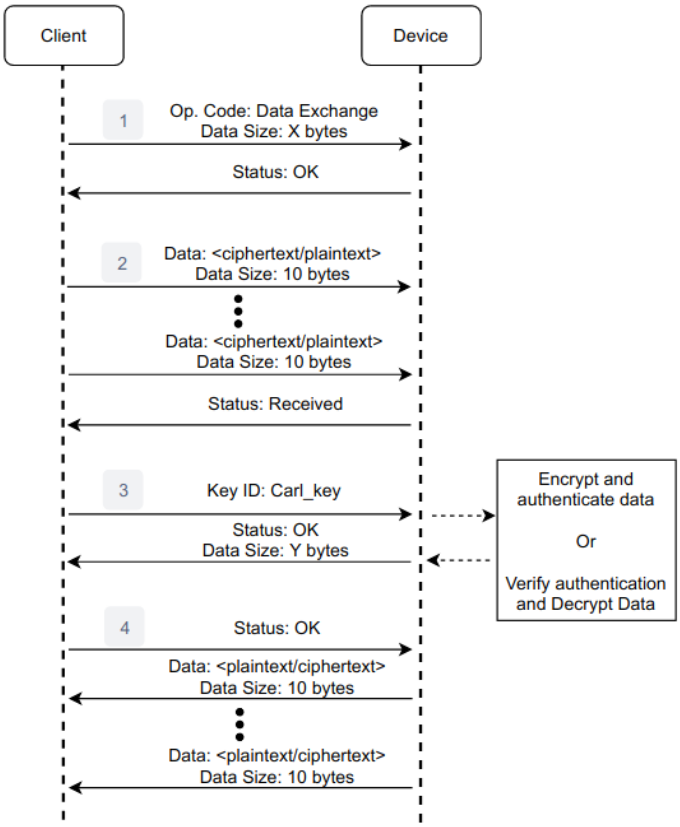

Figure 6.3: Data Exchange Encryption Protocol

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.

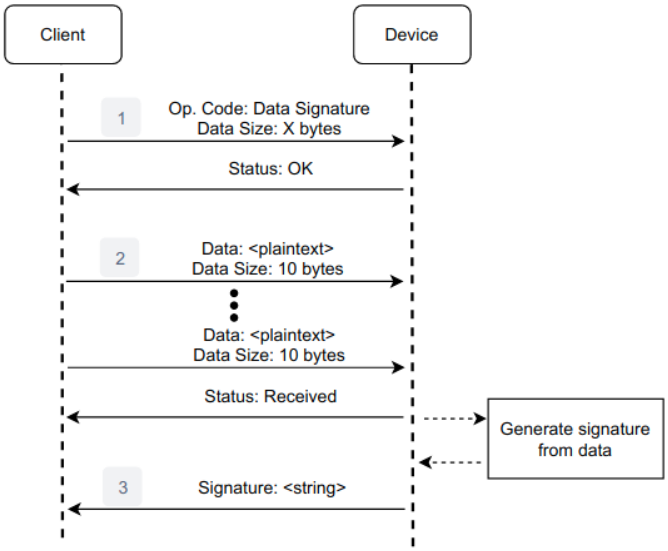

Figure 6.4: Digital Signature Generation

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,

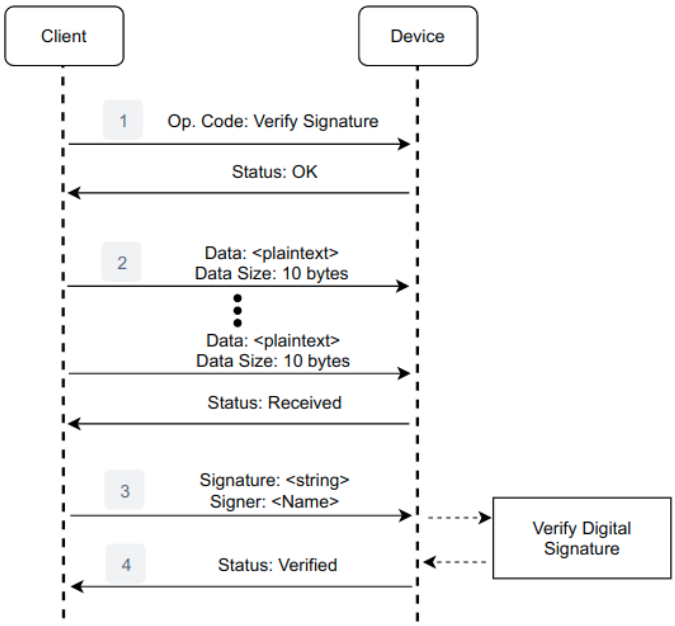

Figure 6.5: Digital Signature Verification

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

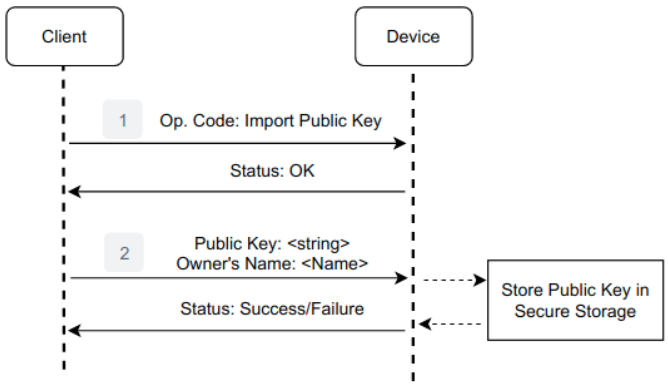

Figure 6.6: Import Public Key

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.

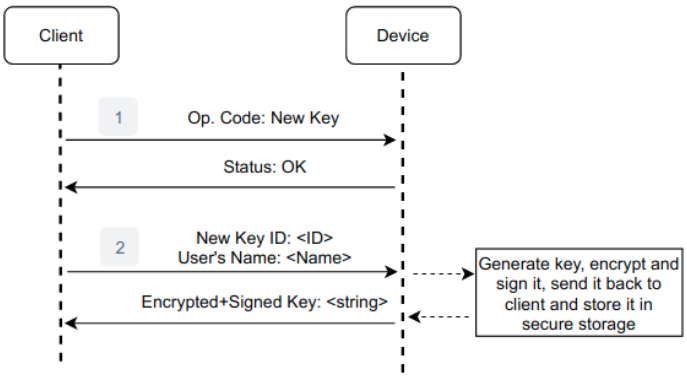

Figure 6.7: Protocol to generate new key to share with user.

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.

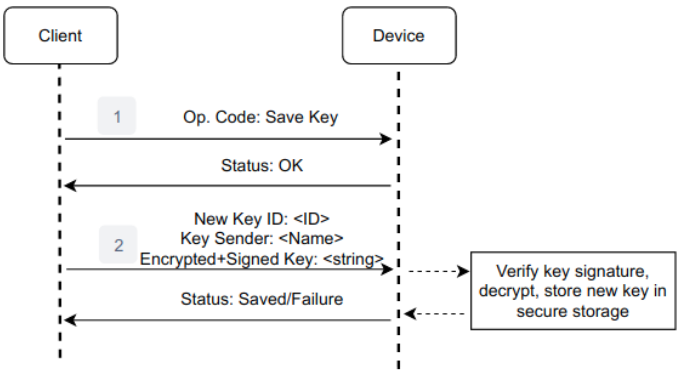

Figure 6.8: Protocol to save key, received from another user.

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.

A

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">
```

```
            <BaseURL>svc_1-L0-</BaseURL>
        </SegmentInfo>
    </Representation>
    <Representation mimeType="video/SVC" codecs="svc" frameRate="30.00" bandwidth="1322.60"
        width="352" height="288" id="L1">
        <BaseURL>svc_1/</BaseURL>
        <SegmentInfo from="0" to="11" duration="PT5.00S">
            <BaseURL>svc_1-L1-</BaseURL>
        </SegmentInfo>
    </Representation>
  </Clip>
</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.

```
; ********************************************************************
; * Constantes
; ********************************************************************

ON     EQU 1 ; contagem ligada
OFF    EQU 0 ; contagem desligada
INPUT  EQU 8000H ; endereço do porto de entrada
           ;(bit 0 = RTC; bit 1 = botão)
OUTPUT  EQU 8000H ; endereço do porto de saída.


; ********************************************************************
; * Stack
; ********************************************************************

PLACE   1000H
pilha:    TABLE 100H  ; espaço reservado para a pilha
fim_pilha:

; ********************************************************************

PLACE   2000H

; Tabela de vectores de interrupção

tab:    WORD  rot0

; ********************************************************************
; * Programa Principal
; ********************************************************************

PLACE   0

inicio:
  MOV BTE, tab     ; incializa BTE
  MOV R9, INPUT    ; endereço do porto de entrada
  MOV R10, OUTPUT    ; endereço do porto de Ìsada
  MOV SP, fim_pilha
  MOV R5, 1      ; inicializa estado do processo P1
  MOV R6, 1      ; inicializa estado do processo P2
  MOV R4, OFF      ; inicializa controle de RTC
  MOV R8, 0      ; inicializa contador
  MOV R7, OFF      ; inicialmente não permite contagem
  EI0           ; 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 LaTeXin it:  πx^2
4      end
5      [:,ind] = max(vec);
6      x_last = x(1,end) - 1;
7      v(end);
8      ylabel('Voltage (μV)');
9  end
```

[1] MATLAB 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`

```
%   Copyright 2010 The MathWorks, Inc.
function ObjTrack(position)
% #codegen
% First, setup the figure
numPts = 300;              % Process and plot 300 samples
figure;hold;grid;          % Prepare plot window
% Main loop
for idx = 1: numPts
    z = position(:,idx);   % Get the input data
    y = kalmanfilter(z);   % Call Kalman filter to estimate the position
    plot_trajectory(z,y);  % Plot the results
end
hold;
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

```
<!DOCTYPE html>
<html>
  <head>
    <title>Listings Style Test</title>
    <meta charset="UTF-8">
    <style>
      /* CSS Test */
      * {
        padding: 0;
        border: 0;
```

```
        margin: 0;
      }
    </style>
    <link rel="stylesheet" href="css/style.css" />
  </head>
  <header> hey </header>
  <article> this is a article </article>
  <body>
    <!-- Paragraphs are fine -->
    <div id="box">
      <p>
        Hello World
      </p>
      <p>Hello World</p>
      <p id="test">Hello World</p>
      <p></p>
    </div>
    <div>Test</div>
    <!-- HTML script is not consistent -->
    <script src="js/benchmark.js"></script>
    <script>
      function createSquare(x, y) {
        // This is a comment.
        var square = document.createElement('div');
        square.style.width = square.style.height = '50px';
        square.style.backgroundColor = 'blue';

        /*
         * This is another comment.
         */
        square.style.position = 'absolute';
        square.style.left = x + 'px';
        square.style.top = y + 'px';

        var body = document.getElementsByTagName('body')[0];
        body.appendChild(square);
      };

```

```
      // Please take a look at +=
      window.addEventListener('mousedown', function(event) {
        // German umlaut test: Berührungspunkt ermitteln
        var x = event.touches[0].pageX;
        var y = event.touches[0].pageY;
        var lookAtThis += 1;
      });
    </script>
  </body>
</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

```

@media only screen and (min-width: 768px) and (max-width: 991px) {

  #main {
    width: 712px;
    padding: 100px 28px 120px;
  }

  /* .mono {
    font-size: 90%;
  } */

  .cssbtn a {
    margin-top: 10px;
    margin-bottom: 10px;
    width: 60px;
    height: 60px;
    font-size: 28px;
    line-height: 62px;
  }
```

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

B

A Large Table

Aliquam et nisl vel ligula consectetuer 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, consectetuer 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, consectetuer et, lacinia sit amet, quam. Suspendisse wisi quam, consectetuer 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, consectetuer 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, consectetuer 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 consectetuer. 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)