TrustZone-backed Bitcoin Wallet

Miraje Gentilal
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
http://sips.inesc-id.pt/
miraje.gentilal@ist.utl.pt

***Abstract*—The popularity of virtual currencies has been increasing at a steady pace in the recent years, largely due the growing acceptance of Bitcoin, whose core is heavily based on public key cryptography. While the public key is used as an address for a payment, the private key is used to access the funds stored in the correspondent address, through the digital signature of transactions.**

Sensitive information is processed in an environment susceptible to attacks in almost every Bitcoin wallet that is currently available. Moreover, the security that is provided to store such information, is supported on the encryption with a key based in a user's password, making it susceptible to dictionary attacks. As the private key is essentially what defines the ownership of bitcoins, it must be very well protected and secured. Thus, the need of using stronger security measures than the ones that are currently used arises.

ARM has made available an extension of processors architectures, designated TrustZone, which allows for the separation of trusted and non-trusted environments, while ensuring the integrity of the Operating System code.

In this dissertation, it is proposed the exploitation the ARM's technology, in order to implement a flexible and reliable Bitcoin wallet that is more resilient to dictionary and side-channel attacks. Making use the TrustZone comes with the downside that writing and reading operations become slower, due to the encrypted storage, but it is shown that cryptographic operations can in fact be executed more efficiently as a result of platform-specific optimizations, which are available through the use of a TrustZone API.

***Index Terms*—Bitcoin; Bitcoin Wallet; Hardware Wallet; Secure World; Trusted Execution Environment; TrustZone.**

I. Introduction

In the last few years, bitcoin has been increasing in popularity in the financial and business sectors. Bitcoin is considered to be the first one of its kind, it uses a peer-to-peer network to generate and exchange digital currency, as well as to verify all the transactions [1], [2]. Since the users have full control of their money, there are several advantages in using this innovative currency, such as low fees and irreversible transactions, that do not reveal any personal information. Moreover, transactions can be verified by everyone through a public ledger leading to a stronger protection against frauds [3].

Every transaction makes use of public key cryptography. To make transactions, a user needs to be able to generate key-pairs and store private information. Private keys are used to sign transactions, and consequentially spend the funds stored in the corresponding public keys, also known as bitcoin addresses. Since the private key is essentially what defines the ownership of bitcoins, it must very well protected and secured.

TrustZone is a technology that is used as an extension of processors and system architectures to increase their security [4]. It enables a logical separation between a secure and a non-secure environment, and when combined with a well designed system and secure software it can ensure that sensitive data remains safe, no matter what the non-secure environment does [4]. These technological characteristics make it very attractive for certain payment services and it has been considered as a means to improve the security and the privacy of users in related art [5]–[8]. These include a platform framework that can be used for applications that require a privacy preserving online remote prepaid payment system [5] and a mobile receipt system that offers security both for customers and merchants [6]. TrustZone has also been considered for the development of Bitcoin mobile wallets, but no experimental results or security analysis has been provided [7], [8].

As bitcoins are considered the "*next logical step in financial operations*" [9], the usage of bitcoins will become increasingly commonplace, creating the need to use stronger security measures than the ones that are currently used. For example, using a user's password, which can be easily guessed, to encrypt private information has proven to be a weak-point for bitcoin wallets [10]. Also, the process of signing transactions should be executed in a secure environment, where the use of the private key is quite safe. While sensitive data can be stored or transmitted in a protected form, its processing is usually done in the clear, which means that it can easily be exposed to an attacker, leading to catastrophic results. With this in mind, in this dissertation the TrustZone technology will be exploited to develop a more flexible hardware-wallet to increase the security behind the generation of keys, their storage and the process of signing each transaction. Furthermore, this wallet is made available as open-source[1].

II. Background

In this section the different types of available bitcoin wallets are reviewed as well as the TrustZone technology.

A. Bitcoin Wallets.

With bitcoin, ownership of currency is maintained through a public ledger, denoted blockchain, that is stored in a peer-to-peer network. Transactions allow to change the ownership

[1]The final source-code of the developed TrustZone-backed Bitcoin wallet will be made publicly available on the 21st of October (due to procedures related with the submission of a paper for the ICISC conference), and in the meantime can be accessed in [11].

of bitcoins, and are supported on public-key cryptography, using digital signatures. These are usually handled by bitcoin wallets, whose main functions are: i) to create and store new addresses and their corresponding private keys; ii) show the balance of each address; and iii) sign transactions. There are many types of bitcoin wallets, as presented in Table I, and, as such, some of the main functions may be disabled due to their nature. Every type of wallet can be categorized as a *hot wallet* or a *cold wallet*. Whereas hot wallets are meant to be used on a daily basis, and thus are typically connected to the Internet for a faster and more convenient use, cold wallets remain offline for less exposure and are used for long-term storage of large amounts of bitcoins.

For example, a *light desktop wallet* is a hot wallet as it needs to access the blockchain for everyday use. Hot wallets are not very secure, since their connection to the Internet makes them quite susceptible to a variety of attacks. On the other hand, the *paper wallet* corresponds to a cold wallet, because it is just a piece of paper where a key-pair is printed. Cold wallets are very secure but lack on usage flexibility, and thus are more recommended for storing large amounts of bitcoins. The *hardware wallet* is a cold wallet that has the best security/functionality ratio of all the wallet types [12]. It is a dedicated piece of hardware that provides increased security features, and it is capable not only of storing private-keys in the long-term but also of signing transactions offline.

TABLE I
MAIN TYPES OF BITCOIN WALLETS.

Type	Description	Characteristics	Most Known Wallets
Bitcoin Clients	The host of the wallet becomes a full node of the Bitcoin network, which means that the whole blockchain is downloaded and kept locally.	- Initial setup takes a long time due the large size of the blockchain; - Private keys are stored on the host giving more control over the bitcoins.	- Bitcoin Core [13].
Light Desktop Wallets	Operates in a similar manner as the Bitcoin Clients but it does not download the full blockchain.	- Setup is relatively fast; - Private keys are stored on the host giving more control over the bitcoins.	- Multibit [14]; - Electrum [15]; - Armory [16].
Online Web Wallets	A third party stores the private keys in their servers which are accessible through an account created for their services.	- Entrusts the security to the service provider who has full control over the client's bitcoins; - Flexible as most of them have mobile and desktop versions available.	- Coinbase [17]; - Xapo [18]; - blockchain.info [19].
Mobile Wallets	Used in mobile devices. It could be an application that works as a Light Desktop Wallet or as an Online Web Wallet.	- The most convenient wallet; - Can have enhanced features: use of QR codes and NFC.	- Bitcoin Wallet [20]; - Mycellium [21]; - Copay [22].
Physical Wallets / Paper Wallets	A piece of paper that has the private and public keys printed on.	- Can easily be generated from different websites; - Does not allow to consult the balance on their own; - Could be very secure if well protected.	Most used website to generate this type of wallet: https://www.bitaddress.org/
Hardware Wallets	A secure hardware device is responsible for the generation and storage of keys and the signing process.	- Very secure; - Does not allow to consult the balance on their own; - Resistant to computer viruses.	- Ledger Nano [23]; - KeepKey [24]; - Trezor [25]; - BitSafe [26].

B. TrustZone Technology

TrustZone is a technology developed by ARM that aims to establish trust into any platform that uses its architecture, which is widely deployed especially among smart devices platforms. This technology takes advantage of the internal hardware that is located within the System on Chip (SoC) along with software components to create a secure and trusted environment where confidentiality and integrity of almost any asset can be guaranteed [4], [27].

TrustZone relies on two distinct concepts of worlds: the *Secure World*, which includes the trusted software, hardware and data; and the *Normal World* which is the common execution environment where other applications run. Both of these worlds are executed in parallel and the system guarantees that no secure component can be accessed outside the trusted environment. It is thus possible to have Trusted Applicationss (TAs) that can access secure memory space and peripherals with the guarantee that the provided data was not tampered with and is trustworthy [4]. Furthermore, TAs can write data to persistent storage in an encrypted manner, using a key that is only available to it through the Secure World. These applications can be implemented using platform-independent Application Programming Interfaces (APIs), which were standardized by GlobalPlatform [28]. GlobalPlatform is an industry association that has been responsible for driving Trusted Execution Enviornment (TEE) standardization releasing APIs such as: *TEE Client API Specification* that enables the communication between normal applications and secure applications; *TEE Internal Core API Specification* that is used by the TAs to perform general security functions, and *Trusted User Interface API Specification* that allows the secure applications to safely display text and graphics [28].

There are available multiple implementations for the Operating Systems (OSs) that run on the Secure and Normal worlds. Open Source Trusted Execution Enviornment (OP-TEE) [29] provides both of them. In Fig. 1 it is presented the software architecture of the OP-TEE as an example of a possible implementation. The processors that implement the TrustZone technology use its integrity checking functionality to run a trusted boot and to create an hardware isolation which in combination with a trusted OS make up a TEE. Typical use cases of this environment include the protection of authentication mechanisms, cryptography, key material and Digital Rights Management (DRM) [30]. When a *Security Client* wants to perform an operation available on a *Security Service*, it firstly invokes a command using the *Client API* which in turn will use the *OP-TEE Driver* to communicate with the *Secure Monitor* that switches to the Secure World. Then, the *TEE Core* can use the *Internal API* to call the desired functions on the Security Service which will perform the operation. The obtained result is returned to the user space following a similar process. Moreover, it is important to notice that every TA has its own private memory space that cannot be accessed neither by normal world applications nor other TAs.

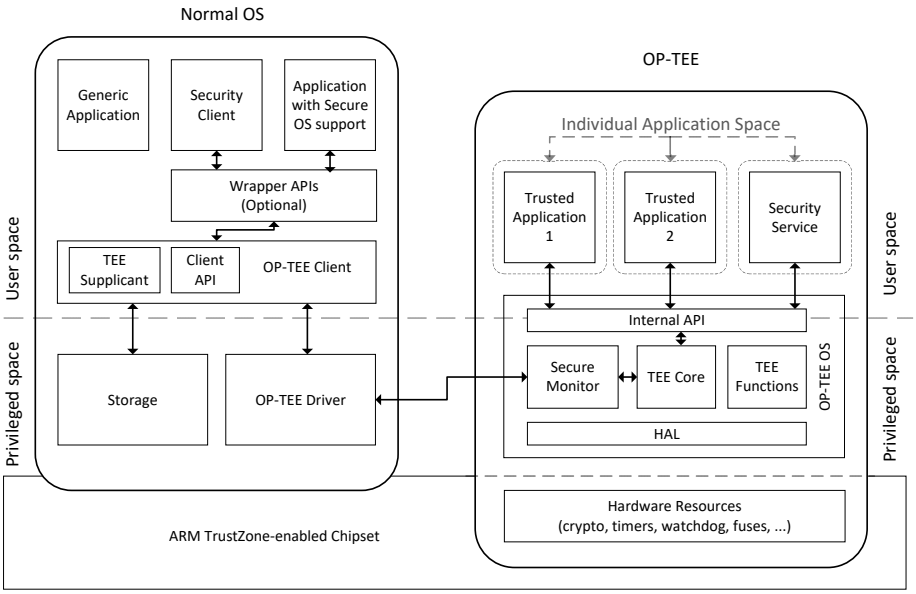

Fig. 1. OP-TEE software architecture [31].

III. Proposed TrustZone-based Wallet

In this section, it is proposed the use of a TrustZone-enabled platform as a wallet. The practical interest of such proposal arises from using the secure storage to store sensitive data, and from taking advantage of the reliable secure cryptographic services provided by the Secure World to perform some of the Bitcoin protocol functions. First, the secure storage prevents certain attacks which are possible to occur on other type of wallets, where a client's password is used to encrypt private information [10]. Since the TrustZone-based wallet encrypts its data using a key that is only available at the hardware level, no dictionary attacks are possible. The usage of the cryptographic services provided by the OS, that were developed by cryptographers with high expertise, also mitigates possible side-channel attacks. Another benefit of this configuration is the integrity check that can verify if a piece of code of the wallet has not changed and verify if the OS is unaltered before booting, preventing, in this way, any unauthorized or maliciously modified software from running [4]. This allows the possibility of running multiple sensitive applications on the same device, such as those used for home banking, since they will not interfere with each other.

In the following subsections, a base wallet is described, and it will be shown how the proposed techniques improve it.

A. Base wallet

The main types of wallets were already discussed in subsection II-A and by analyzing Table I there are two types that stand out from the others: i) *Mobile Wallets* due their usage flexibility and complete functionality; ii) and *Hardware Wallets* due their increased security. As one of the main purposes of this dissertation is to create the most possibly secure wallet, while still enabling for extended flexibility, an hardware wallet have been expanded by porting it to a TrustZone-enabled system. While the prosed wallet does not provide the possibility to check the balance on its own, it can be easily expanded to check the blockchain through a hosting computer. The need to carry an hardware device thus becomes a minor trade-off for the level of security that is provided.

All the examples of *Hardware Wallets* given in Table I were effective candidates as a base for development, but BitSafe was the most promising one: it is fully Open-Source including all manufacturing files and is platform-independent. However, the final version is not yet available in the market, nevertheless there is a great interest in the product from the users side [10], [32]–[34]. Thus, in this work the development prototype will be considered, which was greatly discussed and disclosed in a Bitcoin related blog [33] by its original creators. The main characteristics of this development wallet are presented in Table II.

B. Proposed improvements

The methodology followed for the implementation of BitSafe in a TrustZone-enabled system consisted in porting the source code of the Hardware Wallet, which was originally programmed in a bare-metal way to an OS. Then it is analyzed how the wallet was structured, how it operates and how it could be improved using TrustZone. Lastly, these improvements are applied by taking advantage of the TrustZone platform.

After the wallet was ported, its contents were analyzed, mainly how the storage is partitioned (Fig. 2) and what are the major modules that it is made of (Fig. 3.a)). From this point on, it is needed to decide which are the most critical parts that should be switched to a more secure environment.

The original wallet storage is depicted in Fig. 2. There are two partitions, one called *Global Partition*, where the general information about the hardware wallet is stored, and another one called *Accounts Partition*, where all the multiple wallets are stored. In turn, each wallet is divided into two sections, one with sensitive data that is encrypted using AES-

TABLE II
BITSAFE'S DEVELOPMENT PROTOTYPE MAIN CHARACTERISTICS [10]

Classification	- Cold wallet; - Hardware wallet.
Functions	- Parse transactions, prompt the user and sign transactions; - Store all the private keys; - Through a basic application it can scan the Bitcoin blockchain and generate transactions [35].
Features	- Possibility of having multiple and independent wallets in one device; - Possibility of having hidden wallets which existence cannot be confirmed without the correct encryption key; - Secure formatting doing a multipass, low-level overwrite with random data; - Deterministic wallet. All addresses are generated from a random seed, so that once a backup is done, no further backups are necessary; - Wallet backups can be encrypted using AES-XEX-128; - All potentially dangerous operations require user confirmation; - When signing a transaction, all output addresses and amounts are displayed, as well as the transaction fee.
Project	- Open source [36] (License: BSD 2-clause [37]); - Platform independent implementation; - Tested on Microship PIC 32 bit MCU [38]; - Tested on Microcontroller NXP LPC11U24 [39]; - Tested on AVR board with ATmega328 chip [40].
Versions	Five versions of this wallet were developed: - Version 1: First prototype [32]; - Version 2: Improved prototype [10]; - Version 3: Development prototype [33]; - Version 4: BitSafe first generation [41]; - Version 5: BitSafe second generation [26].
Original Creators	- Firmware: Chris Chua, known as "someone42" in *bitcointalk.org*; - Hardware: "allten" as it is known in *bitcointalk.org*
Requirements	- Required RAM: 8 kilobytes; - Required non-volatile memory per wallet: 160 bytes.

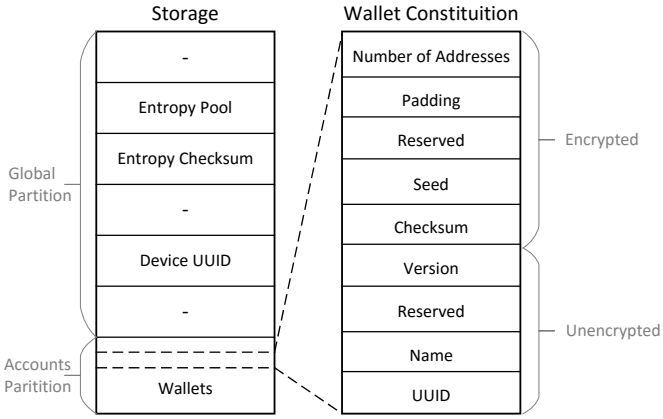

Fig. 2. On the *left* it is displayed how the *Storage* is partitioned, and on the *right* it is shown the information stored in each wallet.

XEX-128, with keys derived from the user's password and a cryptographic salt, while the other is unencrypted and contains general content about the wallet. Despite of the most critical data being encrypted, this section is susceptible to dictionary-based attacks, where an attacker tries a large amount of the most typical passwords for decrypting data, until he finds a suitable one.

In the proposed system, there is an extra layer of security and protection, which was implemented by converting the storage to the Secure World. However, it was noticed that whenever one would write or read from this storage, the whole disk sector would have to be decrypted, leading to a very large overhead. Thus, a write-cache was implemented, where intermediate results are stored, and later flushed into persistent storage, to speed-up the write and read operations.

As previously explained, it is important that the amount of code that runs on the secure-world should be minimum, so that it can be easily reviewed, and the quantity bugs is minimized. Therefore, mostly the OS-provided APIs were used for the implementation of the wallet cryptographic operations. Regarding the *General Wallet Functions* in Fig. 3.a), only the most critical ones were passed to the Secure World, namely those that deal with sensitive data, as a way not increase too much the size of the TA. For example, the *Wallet Functions* related to getting and setting keys were only handled inside the secure world, to completely remove the possibility of anyone getting the private key. Furthermore, the processes for creating new wallets, backing them up as well as creating new addresses were restricted to the Secure World. For the *Transaction Functions*, the most important ones, such as signing, were converted to the Secure World as they have the most impact in the whole wallet. Finally, the *Random Number Generation Functions* exploited the TrustZone API to obtain cryptographically strong random numbers, which is an important requirement in several cryptographic implementations.

The final configuration is presented in the Fig. 3.b). The proposed changes ensure that the handled sensitive data and code cannot be tampered with by any other application in the same device allowing multiple security-sensitive applications to run simultaneously. The proposed system is thus more flexible than most hardware-wallets, since it can be used simultaneously for other applications, such as those used for home-banking, user authentication and trusted processing.

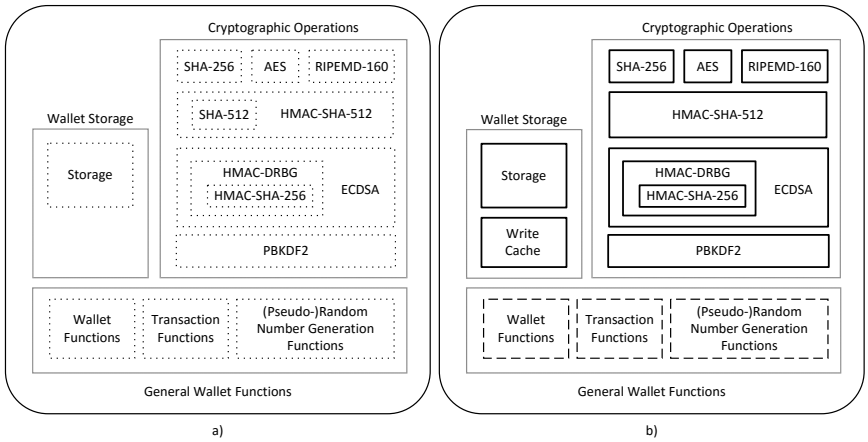

Fig. 3. BitSafe's main modules. *a)* Initial configuration. *b)* Final configuration. The *dotted boxes* are fully implemented in the Normal World, the *dashed boxes* are partially implemented in the Secure World and the *full boxes* are completely implemented in the Secure World. Some of the *Cryptographic Operations* are only used to implement the functions of the *boxes* they are contained.

IV. Implementation Details

In this section, it is explained in more detail the implementation of the proposed wallet. A platform with different characteristic than the ones where most wallets are deployed was used, because those do not support the TrustZone or do not allow the development of TAs, since this is blocked by the Original Equipment Manufacturers (OEMs).

In the specific case of BitSafe, as it was mentioned in section III-A, the chosen base wallet is a prototype that was released a few years ago, and it is not know exactly which hardware was used. Furthermore, as there is no information about the processor that was used, even if there was the possibility of obtaining it there would be no guarantee that would support TrustZone. Nevertheless, as all the code is publicly available and there is the possibility of modifying it, since the underlying license allows it, the base wallet can be deployed in a different platform whose processor supports the TrustZone technology. Therefore, the next step is to choose a platform for that same end. There are few development boards that allow the use of the TrustZone to create TAs [42]–[45]. All of these boards are suitable since they fulfill the minimum requisites imposed by the wallet. For the development of this dissertation, the ARM Juno-r2 board was used [45], which has an ARMv8-A big.Little architecture, with a 1.2 GHz dual-core Cortex-A72, and a 850 MHz quad-core Cortex-A53, with 8GB of DDR3L RAM. This platform was created by ARM with the intent of offering hardware and software developers a chance to work on next-generation hardware with their latest designs. Since these processors represent the future consumer devices, they will ultimately be used for bitcoin wallets, and thus are most relevant for this work.

The OP-TEE, as previously referred, provides an implementation of the OSs of both Normal and secure Worlds for several development boards, including the ARM Juno board, and also has support for GlobalPlatform TEE APIs. For the Normal World, *BusyBox* is used as the operating system, since it was mainly developed to be used on embedded devices with low computational power. In the secure world, that role is fulfilled by the *OP-TEE OS* whose size is about 244 KB. It is important to note that the code for the secure OS should be as small as possible, since it comprises the trust anchor, and thus should be easily reviewed and have the minimum attack surface.

As the *OP-TEE OS* supports the GlobalPlatform specifications, these APIs were employed [28] (corresponding to functions prefixed with *TEE*). The secure storage was implemented using a persistent object for all the partitions in Fig. 2. Meanwhile, the write-cache consists in a buffer with the size of a disk sector. Whenever there is data to be written, it is checked if the correspondent disk sector is already available in the cache. If it is not, the disk sector is loaded into the write-cache and the previous content is flushed. For the reading process, if the requested data is available on the write-cache, then it is read from there; otherwise the data is read directly from the data stream. To write data into the data stream the *TEE_WriteObjectData()* function is used and the *TEE_ReadObjectData()* function is used ro read data streams.

For cryptographic operations, the API requires an operation handler obtained with *TEE_OperationHandle*, which is used as an argument for the subsequent function calls. The Secure Hash Algorithm with 256 bits (SHA-256), Advanced Encryption Standard (AES) and Hash Message Authentication Code with SHA-256 (HMAC-SHA-256) or with SHA-512 (HMAC-SHA-512) are multi-staged operations, in which one function is used to initialize the operation, another to feed it with data and another to finalize it. For SHA-256, *TEE_DigestUpdate()* and *TEE_DigestDoFinal()* were applied (there is no initializer function); for AES, *TEE_CipherInit()*, *TEE_CipherUpdate()* and *TEE_CipherDoFinal()* were used; lastly, for HMAC with SHA-256 and SHA-512, *TEE_MACInit()*, *TEE_MACUpdate()* and *TEE_MACComputeFinal()* were used.

OP-TEE also provided implementations for the Password-Based Key Derivation Function 2 (PBKDF2) and Elliptic Curve Digital Signature Algorithm (ECDSA) functions but the former was an extension of the GlobalPlatform API, so other secure OSs might not have it, whereas the latter one is only supported by a few implementations of the secure OS. Since the wallet was designed to be as platform independent as possible, while still providing high levels of security, these

operations were manually extended to the secure world. For the RACE Integrity Primitives Evaluation Message Digest with 160 bits (RIPEMD-160) function, as it was not available in any API (nor it was specified by GlobalPlatform), its source code was also ported to the Secure World. The same rationale was applied to the HMAC Generation Using Deterministic Random Bit Generator (HMAC-DRBG).

The *General Wallet Functions* were developed by porting the corresponding original code to the secure world making the necessary adaptations and using the previous cryptographic functions when required. The communication was implemented using an *TEEC_TempMemoryReference*, which pointed to a buffer allocated in the normal world, where the secure world would write the function results.

V. Experimental Results

In this section, the performance of the main *Cryptographic Operations*, *Storage* and *General Wallet Functions* of the developed wallet is assessed and compared with the original code ported from [36] into the ARM Juno-r2 board. We use the term TrustZone-backed Bitcoin Wallet (TBW) to denote the proposed wallet while Original Base Wallet (OBW) is used for the original one. The configurations adopted for comparison are represented in the Fig. 3. All the execution times presented in this section correspond the median of the number of clock cycles of 100 executions. These times were obtained using the *clock()* function from the C native library. All TW calls include the overhead of the communication with the Secure World.

In Table III, it is visible that there was an increase in the number of the execution time for the storage functions, when comparing the TBW with the OBW, but this was expected as the secure storage has an extra layer of encryption provided by the secure world [46]. When the secure storage is used directly, the overhead of the additional encryption process has a significant larger impact. After adding a write-cache, the effect of the overhead is greatly mitigated if the requested block is already in memory; otherwise, one has to update the cache, with the associated overhead of reading from or writing to the secure storage.

In Table IV, it is presented the performance of the most frequently used cryptographic functions in the wallet, where the SHA-256 was the only negatively affected operation for being executed in the Secure World. However, it is important to note that the execution time of this function in the TBW is practically independent of the data size that is being processed, making this operation more robust against side-channel attacks. The AES-XEX-128 function was proven to be much faster when the API provided by OPTEE for the Secure World was used. This confirms that the use of standardized APIs is beneficial to obtain the maximum performance, since each vendor is able to implement the functions in the most optimized manner for the underlying hardware. The ECDSA and PBKDF2 where both manually implemented in the secure world, and since they operate "closer" to the hardware, their execution time has also improved largely.

In Table V, as it was expected, all the functions that directly involve reading or writing to the secure storage had its execution time increased. The function to *Format the storage* was the most impacted as it performs multipass, low-level overwrites with random data in the full size of the wallet to guarantee that its final state is unpredictable, leading to a large overhead from the additional encryption layer. The *Delete wallet* function suffers from a similar issue. On the other hand, the functions that do not directly use the secure storage had its performance improved. These reduced execution times result from the improvements previously described in Table IV. The functions to *Create a new address* and to *Sign a transaction* are among the most important and used ones in the whole wallet. The fact that these functions were able to execute in a faster and more reliable manner confirm the validity of this proposal.

Table VI presents a qualitative comparison between the OBW and the TBW. Firstly, the OBW storage encryption was supported on a password provided by the user. As this type of systems can often be attacked using dictionary-based attacks, the decision to use TrustZone secure storage greatly boosted security by encrypting data using a random key that is only available at the hardware level. While some care had been taken while the OBW was developed regarding side-channel attacks, the OP-TEE API was implemented taking into account the underlying hardware, which allows to provide an higher level of assurance against side-channel attacks. Furthermore, while the OBW used checksums to protect against data tampering, with TrustZone, code is also verified for integrity at boot-time. Moreover, TrustZone allows for the co-existence of multiple TAs, with high assurance that none will interfere with each other. Therefore, it is possible to extend the proposed wallet with other applications that need high levels of security, such as home-banking. Finally, it was seen that while the use of the secure storage deteriorated performance for some functions, the use of the *OP-TEE API* allowed for platform-specific optimizations that enhanced performance for cryptographic operations.

VI. Conclusion

With the increasing use of bitcoins, it is becoming more and more important to have highly secure devices in which private-key information can be safely stored and handled. With this dissertation, it was shown how to greatly increase the security of Bitcoin wallets by using the TrustZone technology. Concretely, a wallet from the related art was analyzed, and it was investigated which functions should be implemented in the Secure World, in order to improve the security of its storage; as well as how to process sensitive information in a more safely manner, using trustworthy and optimized cryptographic operations. From this work a wallet has resulted that is more resistant to a series of attacks, namely dictionary and side-channel attacks, while enabling for a certain flexibility. Furthermore, it was shown that the use of the *GlobalPlatform APIs* allowed for more efficient cryptographic operations, since platform-specific optimizations are used. This reduces the execution time of the most frequently used wallet functions, and compensates for the increased writing and reading times of the encrypted secure storage to attain improved security.

TABLE III
EXECUTION TIME OF STORAGE RELATED FUNCTIONS. THE VALUES ARE PRESENTED IN CLOCK CYCLES. IN PARENTHESIS THE EXECUTION TIME INCREASE IN PERCENTAGE OF THE TBW RELATIVE TO THE OBW IS PRESENTED.

Function	TBW	OBW		
		Secure Storage	Secure Storage + Write-Cache (block not loaded)	Secure Storage + Write-Cache (block loaded)
Write in Global Partition (512B)	34	129111 (+374136%)	106685 (+309133%)	128 (+271%)
Write in Accounts Partition (1024B)	25	129724 (+518798%)	131833 (+527232%)	138 (+454%)
Read from Global Partition (512B)	20	99255 (+496177%)	99485 (+497325%)	128 (+540%)
Read from Accounts Partition (1024B)	6	99680 (+1661241%)	99901 (+1664925%)	136 (+2166%)
Flush data	4	N/A	178734 (+4468262%)	1787345 (+4468262%)

TABLE IV
EXECUTION TIME OF THE MOSTLY USED CRYPTOGRAPHIC FUNCTIONS. THE VALUES ARE PRESENTED IN CLOCK CYCLES. IN PARENTHESIS THE EXECUTION TIME INCREASE IN PERCENTAGE OF THE TBW RELATIVE TO THE OBW IS PRESENTED.

Function	Mode	OBW	TBW
SHA-256	INIT (100B)	2	69 (+3350%)
	WRITE (100B)	43	121 (+178%)
	FINISH (100B)	30	122 (+306%)
	FINISH-DOUBLE (100B)	65	134 (+106%)
	INIT (1B)	2	68 (+3300%)
	WRITE (1B)	2	114 (+5600%)
	FINISH (1B)	35	122 (+250%)
	FINISH-DOUBLE (1B)	70	134 (+91%)
AES-XEX-128	DECRYPT (64B)	794	157 (-80%)
	ENCRYPT (64B)	364	157 (-56%)
ECDSA	SIGN	1280540	190499 (-85%)
PBKDF2	-	203447	34422 (-83%)

TABLE V
EXECUTION TIME OF THE MOSTLY USED WALLET FUNCTIONS. THE VALUES ARE PRESENTED IN CLOCK CYCLES. IN PARENTHESIS THE EXECUTION TIME INCREASE IN PERCENTAGE OF THE TBW RELATIVE TO THE OBW IS PRESENTED.

Packet	OBW	TBW
Format the storage	10373	6900920 (+66427%)
Create a new address	5640665	2159050 (-61%)
Sign a transaction	1348990	296758 (-78%)
Load the wallet	206596	652153 (+215%)
Backup the wallet	108	230 (+111%)
Delete the wallet	3804	2728645 (+71621%)

TABLE VI
COMPARISON BETWEEN THE OBW AND THE TBW.

	OBW	TBW
Resistance against dictionary attacks	✗	✓✓
Resistance against side-channel attacks	✓	✓✓
Code and data integrity	✓	✓✓
Extensibility	✗	✓
Performance	✓✓	✓

REFERENCES

[1] J. Bohr and M. Bashir, "Who Uses Bitcoin? An exploration of the Bitcoin community," in *2014 Twelfth Annual International Conference on Privacy, Security and Trust*. IEEE, jul 2014, pp. 94–101. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6890928

[2] P. Ciaian, M. Rajcaniova, and D. Kancs, "The economics of BitCoin price formation," *Applied Economics*, vol. 48, no. 19, pp. 1799–1815, 2016. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/00036846.2015.1109038

[3] bitcoin.org, "What are the advantages of bitcoin?" "https://bitcoin.org/en/faq#what-are-the-advantages-of-bitcoin", [Online] (Accessed in August 14, 2016).

[4] ARM, "ARM Security Technology. Building a Secure System using TrustZone Technology," *ARM white paper*, p. 108, 2009. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[5] M. Pirker and D. Slamanig, "A framework for privacy-preserving mobile payment on security enhanced ARM trustzone platforms," *Proc. of the 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012*, pp. 1155–1160, 2012.

[6] W. H. W. Hussin, P. Coulton, and R. Edwards, "Mobile ticketing system employing TrustZone technology," *Mobile Business, 2005. ICMB 2005. International Conference on*, pp. 651–654, 2005. [Online]. Available: 10.1109/ICMB.2005.71

[7] Trustonic, "Rivetz Teams with Trustonic to Secure Mobile Bitcoin Wallets", https://www.trustonic.com/news-events/pr/rivetz-trustonic-bitcoin-wallets, July 2014, [Online] (Accessed in August 21, 2016).

[8] J. Young, "Ledger Transforms Smartphones into Hardware Wallets", https://cointelegraph.com/news/ledger-transforms-smartphones-into-hardware-wallets, March 2016, [Online] (Accessed in August 21, 2016).

[9] G. F. Hurlburt and I. Bojanova, "Bitcoin: Benefit or curse?" *IT Professional*, vol. 16, no. 3, pp. 10–15, 2014.

[10] Someone42, "Hardware Bitcoin wallet - a minimal Bitcoin wallet for embedded devices - BitSafe v2", https://bitcointalk.org/index.php?topic=78614.msg1327630#msg1327630, 2012, [Online] (Accessed in September 8, 2016).

[11] M. Gentilal, "TrustZone Backed Bitcoin Wallet Source Code", http://web.tecnico.ulisboa.pt/ist173547/TrustZone-Backed-Bitcoin-Wallet.zip, 2016, [Online] (Accessed in September 28, 2016), Password: m!r@j&D!ss&rt@tion2016.

[12] Bitcoin.org, "Securing your wallet - Bitcoin", https://bitcoin.org/en/secure-your-wallet#hardwarewallet, 2016, [Online] (Accessed in September 22, 2016).

[13] BitcoinWiki, "Bitcoin Core - Bitcoin Wiki," "https://en.bitcoin.it/wiki/Bitcoin_Core", 2015, [Online] (Accessed in September 5, 2016).

[14] MultiBit, "The Bitcoin Wallet for Your Desktop," "https://multibit.org/", 2016, [Online] (Accessed in September 5, 2016).

[15] Electrum, "Electrum Bitcoin Wallet," "https://electrum.org/#home", 2016, [Online] (Accessed in September 5, 2016).

[16] Armory, "Best Bitcoin Wallet Armory — Multi-Signature Cold Storage," "https://www.bitcoinarmory.com/", 2016, [Online] (Accessed in September 5, 2016).

[17] Coinbase, "Bitcoin wallet," "https://www.coinbase.com/", 2016, [Online] (Accessed in September 5, 2016).

[18] Xapo, “Xapo - Bitcoin Wallet & Vault,” ”https://xapo.com/”, 2016, [Online] (Accessed in September 5, 2016).
[19] Blockchain.info, “The World’s Most Popular Bitcoin Wallet,” ”https://blockchain.info/wallet/#/”, 2016, [Online] (Accessed in September 5, 2016).
[20] Bitcoin Wallet developers, “Bitcoin Wallet Application Google Play Store,” ”https://play.google.com/store/apps/details?id=de.schildbach.wallet”, 2016, [Online] (Accessed in September 5, 2016).
[21] Mycelium, “Mycelium Wallet,” ”https://wallet.mycelium.com/”, 2016, [Online] (Accessed in September 5, 2016).
[22] Copay, “Copay Secure, Shared Bitcoin Wallet,” ”https://copay.io/”, 2016, [Online] (Accessed in September 5, 2016).
[23] Ledger, “Ledger Wallet - Ledger Nano - Secure and smart Bitcoin hardware wallet,” ”https://www.ledgerwallet.com/products/1-ledger-nano”, 2016, [Online] (Accessed in September 5, 2016).
[24] KeepKey, “KeepKey: A Simple Bitcoin Hardware Wallet,” ”https://www.keepkey.com/”, 2016, [Online] (Accessed in September 5, 2016).
[25] TREZOR, “TREZOR: The Bitcoin Safe,” ”https://bitcointrezor.com/”, 2016, [Online] (Accessed in September 5, 2016).
[26] Butterfly Labs, “Bitcoin Wallet Hardware — Butterfly Labs,” ”http://www.butterflylabs.com/bitcoin-hardware-wallet/”, 2016, [Online] (Accessed in September 5, 2016).
[27] T. Alves and D. Felton, “Trustzone: Integrated hardware and software security,” *ARM white paper*, vol. 3, no. 4, pp. 18–24, 2004.
[28] GlobalPlatform, “GlobalPlatform Specifications,” ”http://www.globalplatform.org/specificationsdevice.asp”, [Online] (Accessed in August 30, 2016).
[29] Linaro, “OP-TEE : Open Source Trusted Execution Environment,” ”https://www.op-tee.org/,”, p. 2016, [Online] (Accessed in September 26, 2016).
[30] ARM, “Products — TrustZone ARM,” ”https://www.arm.com/products/security-on-arm/trustzone,”, [Online] (Accessed in September 26, 2016).
[31] Linaro, “HKG15-311: OP-TEE for Beginners and Porting Review,” ”http://pt.slideshare.net/linaroorg/hkg15311-optee-for-beginners-and-porting-review,”, 2015, [Online] (Accessed in September 27, 2016).
[32] Someone42, “Hardware Bitcoin wallet - a minimal Bitcoin wallet for embedded devices - BitSafe v1,” ”https://bitcointalk.org/index.php?topic=78614.0,”, 2012, [Online] (Accessed in September 8, 2016).
[33] Someone42, “Hardware Bitcoin wallet - a minimal Bitcoin wallet for embedded devices - BitSafe v3,” ”https://bitcointalk.org/index.php?topic=78614.msg1829506#msg1829506,”, 2013, [Online] (Accessed in September 8, 2016).
[34] Allten, “[ANN] BitSafe Hardware Wallet Now Shipping,” ”https://bitcointalk.org/index.php?topic=152517.0,”, 2013, [Online] (Accessed in September 8, 2016).
[35] Someone42, “bitsafe-example-wallet,” ”https://github.com/someone42/bitsafe-example-wallet,”, 2014, [Online] (Accessed in September 27, 2016).
[36] C. Chua, “hardware-bitcoin-wallet: BitSafe Firmware.” ” https://github.com/someone42/hardware-bitcoin-wallet”, [Online] (Accessed in August 21, 2016).
[37] “BSD 2-clause Simplified License - Choose a License,” ”http://choosealicense.com/licenses/bsd-2-clause/,”, [Online] (Accessed in September 15, 2016).
[38] Microchip, “32-Bit MCUs — Microchip Technology Inc.” ”https://www.microchip.com/design-centers/32-bit,”, 2016, [Online] (Accessed in September 20, 2016).
[39] NXP, “ARM mbed LPC11U24 Board — NXP,” ”http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus/lpc-cortex-m0-plus-m0/lpc1100-cortex-m0-plus-m0/arm-mbed-lpc11u24-board:OM13032,”, 2016, [Online] (Accessed in September 20, 2016).
[40] MikroElektronika, “MINI-AT - AVR USB development board in DIP26 form factor,” ”http://www.mikroe.com/mini/at/,”, 2016, [Online] (Accessed in September 20, 2016).
[41] HardwareWallets.com, “Hardware Wallets - BitSafe v4,” ”http://www.hardwarewallets.com/,”, [Online] (Accessed in September 8, 2016).
[42] ZedBoard.org, “ZedBoard Technical Specifications,” ”http://zedboard.org/content/zedboard-0,”, [Online] (Accessed in September 25, 2016).
[43] 96Boards, “Documentation - 96Boards Hikey Quickstart,” ”http://www.96boards.org/documentation/ConsumerEdition/HiKey/Quickstart/README.md/,”, [Online] (Accessed in September 25, 2016).
[44] NXP, “i.MX53 Quick Start Board — NXP,” ”http://www.nxp.com/products/power-management/pmics/pmics-for-i.mx-processors/i.mx53-quick-start-board:IMX53QSB,”, [Online] (Accessed in September 25, 2016).
[45] ARM, “Juno ARM Development Platform - ARM,” ”http://www.arm.com/products/tools/development-boards/versatile-express/juno-arm-development-platform.php,”, [Online] (Accessed in September 25, 2016).
[46] L. W. Security, “Secure Storage In OP-TEE,” ”https://github.com/OP-TEE/optee_os/blob/master/documentation/secure_storage.md,”, 2016, [Online] (Accessed in September 28, 2016).