A secure structure for generating remote encryption keys

Saeid Azimi[a]

[a]Master of Science, computer engineer, network computer, Iran University of Science and Technology, azimi@cmps2.iust.ac.ir

Article History: Received: 14 July 2020; Accepted: 2 January 2021; Published online: 5 February 2021

Abstract: This paper outlines an ad-hoc architectural design and a practical implementation of a secure, distributed, fault tolerant and scalable infrastructure comprised of a distributed network of electronics hardware systems that remotely generate cryptographic keys, store them, digitally sign cryptographic transactions based on such keys, and record the transactions on a blockchain . The proposed solution is suitable to service both crypto-finance and non-finance applications, and the physical infrastructure is designed to be implemented using off-the-shelf industrial electronics, which further sustains the already scalable-by-design infrastructure architecture.

Keywords: Blockchain, Crypto Assets, Physical Security

1. Introduction

Since the introduction of Bitcoin in 2007, the number of applications based on blockchain patterns and cryptographic assets has grown exponentially. A common requirement for these applications is the use of cryptographic assets, including private keys and derivative concepts such as cryptographic wallets. These digital assets are highly vulnerable to theft, as proven by Bitcoin cases for millions of dollars of theft (Taha Ali, Clarke, 2015). Therefore, maintaining the security of these assets, while maintaining the ability to perform operations such as signing a contract or transferring currency, is vital. Solutions have been proposed for this purpose. Mann and Loebenberger proposed a two-factor authentication system for the Bitcoin protocol (Mann and Loebenberger, 2017). Wu et al. introduced a bitcoin common trading mechanism based in part on illegible fuzzy signatures (Wu et al, 2017). Gentilal et al. instead focused on a protective layer of the system and use the development of new processor architectures for the logo, a designated safe area, which allows the separation of reliable and unreliable environments (Gentilal et al. 2017).

In this paper, we proposed the design of a new distributed infrastructure that aims to safely and securely manage the entire lifecycle of cryptographic assets that play an important role in blockchain applications (Crosby, 2016). Currently, such applications include asset tokenization, formalization of smart contracts, and all foreign exchange operations, including the transfer of funds. We proposed a complete hardware solution for remotely generating cryptographic keys, storing them, digitally signing cryptographic transactions based on such keys, and registering transactions in blockchain.

Such an infrastructure to provide a clear solution to this practical need is not currently met, given the growing population with cryptographic assets (cryptocurrencies and cryptographic wallets), a simultaneous need throughout the system is as follow:

1. Full control, over the entire life cycle of the keys (production, storage and use), over private cryptographic keys that provide unlimited access to users' own cryptographic assets.

2. Perform cryptographic operations remotely while maintaining the security of the underlying keys.

The proposed solution includes system architecture, communication protocol, and key management scheme. A proprietary approach for hardware digital circuits has been developed to ensure that each system's core cryptographic engine stores keys in a physically controlled range and never connects to external networks even when the system is running. It is designed to be isolated from external networks at any time, regardless of system status. This is a significant advantage compared to several other automated solutions available (Ledger, 2018), (Casa, 2018), (Bitgo, 2018), (Trezor, 2018) because their activities are activated to a cryptographic circuit and its connection to the network (s) is restricted.

Research method

In this section, the research method, statistical population, sample and sampling method, research tools (how to check the validity and reliability of tools) and data analysis methods (non-research articles are excluded from this framework) were investigated. Currently, the only part that needs to be met, or at least partially met, is cryptographic

finance. In particular, cryptographic tokens and cryptocurrency payment investment programs are considered as a subset or specific type of its general category called cryptographic assets (Conley et al, 2017). Systems designed for cryptographic financing typically manage only a specific subset of cryptographic operations (Gennaro et al, 2016).

Now solutions in this regard are suitable for the customer, purse hardware, end users allows transactions using currencies or password currencies (Casa, 2018), and currencies or symptoms are tracked in their portfolio. They mostly work independently, and the most commonly used of these solutions is an individual end user who uses a hardware wallet connected to proprietary software.

These features include storing beads and cryptographic keys on the physical dongle, while wallet management, that is, managing various financial situations related to cryptocurrencies, is done by software running on a smartphone or PC. In the commercial sector, the implementation of a number of hardware security modules (HSMs) (Mavrovouniotis and Ganley, 2014) is dedicated to the cryptographic sector, such as that produced by the General Office.

However, from the analysis of publicly available documents, such modules do not give end users complete and direct control over encrypted private keys (Hofmann et al, 2018). There are also a number of wallets that have not yet been implemented. Recently, Chen et al. proposed a blockchain-based payment collection monitoring system using a comprehensive digital bitcoin wallet (Chen et al, 2017). The implementation of the old wallet is called the old wallet by Bamert et al. (Bamert et al, 2014). This device can also be used as an e-wallet in combination with a point of sale and acts as an alternative to cash and credit cards.

However, none of the proposed solutions offer a distributed architecture for storing and managing cryptographic assets. In particular, a distributed privacy solution in this area is not proposed for other distributed systems (Zeilemaker et al, 2013). In this article, we address this problem with distributed architectural design.

Our method allows remote control and storage of cryptocurrencies and private keys wallets while ensuring a high level of access to increase communication security and create an additional layer of privacy, a hybrid-style network with layered cryptography is used (Palmieri and Pouwelse, 2014) (Güneysu et al, 2014).

Findings

In this paper, we proposed an architecture that physically isolates key cryptographic components and information (e.g. keys) to provide security against enemies that access the device remotely. The proposed architecture is described in the prototype implemented in Section 4.

The infrastructure we propose combines layered cryptography modules with hardware security and consists of three system classes consisting of: user devices, platform nodes (also called relays), and nodes. The capabilities of each class of nodes as well as the information they have are described in Table 1.

User Ui devices are designed to be small portable devices that can perform simple operations when connected to a host computer. Their role is to authenticate the user and be able to communicate with operating system nodes. Each user has a device. The device itself stores the elliptic curve of the public / private key pair of elliptic curves (PkUi, SkUi) based on curve 25519. (Bernstein, 2006). Through the software that is installed on the host computer, the user device can use ECC and key pair as well as symmetric cryptographic salsa 20 (Bernstein, 2006) to establish a cryptographic connection. We note here that attacks on the host computer are beyond the scope of this article and are not considered in the analysis.

Pi platform nodes and platform nodes act as relays between user devices and remote nodes. Their role is to disconnect nodes from the Internet and provide access control in user accounting and business billing implementations. Depending on the size of the overall infrastructure, there are a number of relays used by commercial cloud solutions on the market. It also protects nodes from denial of service attacks. Each node has a pair of public / private elliptic curve keys and is able to create a cryptographic connection using ECDH and the salsa 20 symmetric cryptographic. In platform nodes, it stores the number of times by each remote node for later use to create a key agreement with an elliptic curve. The node is signed with its own key by the generated node and stored by each remote node for later use in the timestamp in creating the Duffy Hellmann Elliptic Curve Key Agreement (ECDH) and the secondary nodes by the manufacturer node with the private key is signed.

Di remote nodes store several remote nodes collectively in cryptographic assets (such as private keys) that are protected by infrastructure. In order to create an insulated surface, assets are stored on a dedicated device that is separate from the main remote node system and connected to the Internet, although protected by a firewall that allows only input nodes from the platform to connect. It is assumed that the proprietary electronic device will remain in most cases and will be activated only if you need to perform the relevant cryptographic operation according to the instructions of the user devices.

A dedicated electronic device is intended to remain on the Internet most of the time and will only be activated if the cryptographic operation is performed in accordance with the user device instructions. This separate cryptographic device is designed to produce and store cryptographic keys. Because all relevant cryptographic operations can be performed internally for the node, the keys are never exposed to the main node system as well as to the Internet world. In order to generate strong keys, the device embeds a real-time random number (TRNG) subsystem.

The relationship between the parties is described in Figure 1:

1. The first step is taken by the Ui user, who begins his communication by performing an elliptical Huffman curve. The Key Agreement (ECDH) starts with a selected Pi platform, where the user can freely choose from existing nodes. The agreed key is then used for cryptography of any further communication between the two parties using salsa 20. After establishing the cryptographic connection between the Ui and Pi, the user declares a Di that he wants to perform the required operation and announces his choice to the platform node. The Pi platform node stores the default nans Di in the cache.

A Nonce is selected and send it to the Ui. The user can verify the randomness and integrity by checking the Di signature on its public key. When a key is agreed upon by the Ui and Di, a cryptography tunnel is created between them using salsa 20.

However, the connection will still be made by Pi, because Ui cannot communicate directly with Di. The Ui source packets are cryptographed twice and use layered cryptography in a similar way to onion routing (Syverson et al, 1997).

3. The channel between Ui and Di is used to transmit the performed cryptographic instructions, transmitting the results to the user using the cryptographic assets stored in the remote node if necessary. Note that Pi and Di are also created by a cryptographic tunnel created with the same mechanism as Ui and Pi: this part of the protocol has been removed from Figure 1 for simplicity.

Table 1: Functions of each system component

Devices	Keys and Information	Main Functions	HW Details		Web Interfaces

U_i	sk_{U_i}
	pk_{U_i}
	$\mathsf{pk}_{P_{1..n}}$
	$\mathsf{pk}_{D_{1..n}}$
		`Curve 25519`
		Signatures
		Verification
		`Salsa20`
			microcontroller
			display
			single button
			self-installing SW for
			interfacing with PC		

P_i	sk_{P_i}
	pk_{P_i}
	$\mathsf{pk}_{U_{1..n}}$
	$\mathsf{pk}_{D_{1..n}}$
	Ordered cache of $D_{1..n}$ nonces
	with expiration time
		`Curve 25519`
		Signatures
		Verification
		`Salsa20`
		`Lamport Signatures`
			HW module connected to
			server via USB
					`ssh`

D_i	sk_{D_i}
	pk_{D_i}
	$\mathsf{pk}_{U_{1..n}}$
	$\mathsf{pk}_{P_{1..n}}$
	Asset to be protected
		`Curve 25519`
		Signatures
		Verification
		Key generation
		ECDSA
		`Salsa20`
		`Lamport Signatures`
			TRNG
			HW cryptography
			Shared memory
			Connected Part
			`Curve 25519`
			Verification
			`Salsa20`
				Not Connected Part
				`Curve 25519`
				Verification
				`Salsa20`
				TRNG
				ECDSA
					`ssh`

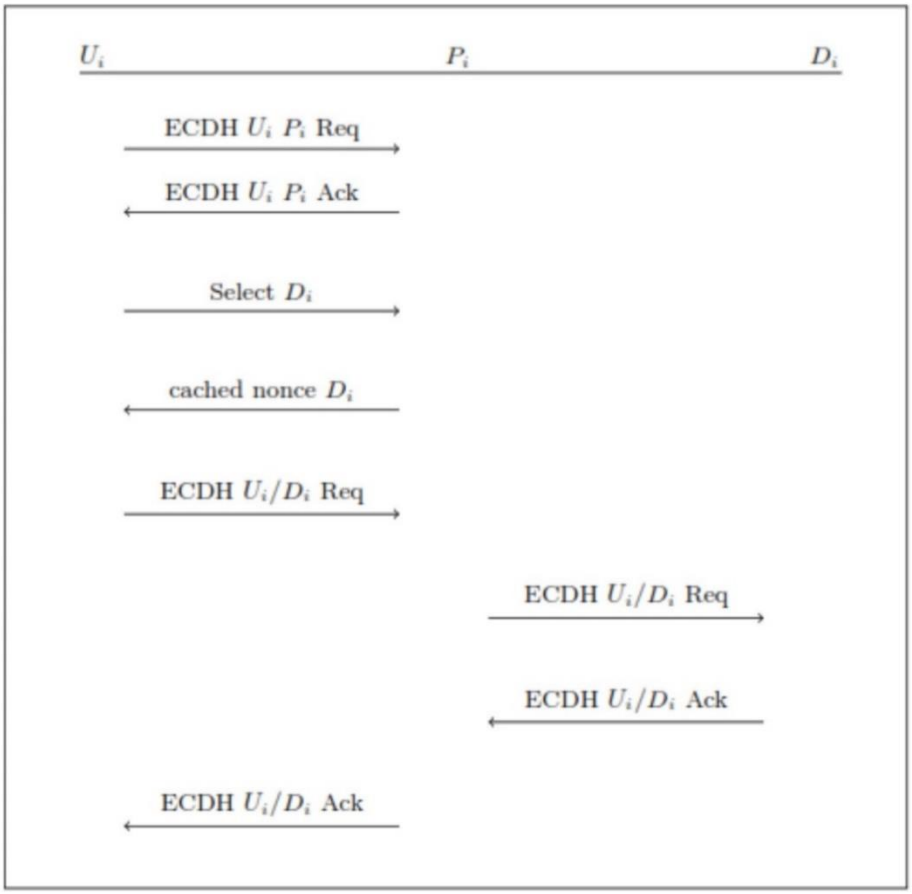
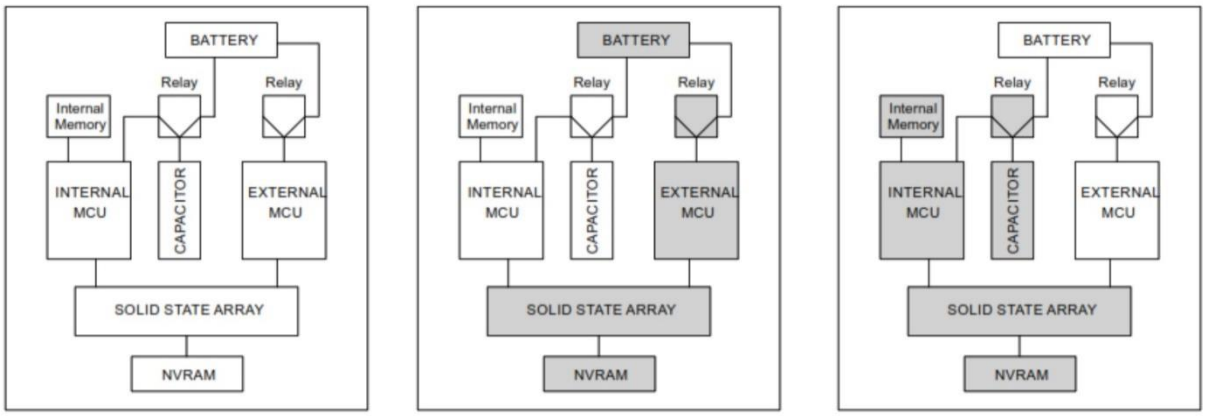


Figure 1: The layered cryptography protocol is used to create a secure tunnel between user U devices and remote D nodes, using P-platform nodes as transparent relays.

Performs the required operations and announces its selection to the platform node. A platform node pre-calculated by di selects the ones to be saved and sends it to the user interface. The user can verify the randomness by checking the Di signature in front of its public key. This nonce allows Ui and Di to perform the Duffy Hellmann Elliptic Curve Key Agreement using Pi as the relay. When a key is agreed upon by Ui and Di, a cryptographic tunnel is created between them using salsa 20. However, the connection will still be made by Pi, because Ui cannot communicate directly with Di. Ui source packets are cryptographed twice and use layered cryptography in a manner similar to so-called onion routing (Syverson et al, 1997). The channel between Ui and Di is used to transmit cryptographic instructions, which is done using cryptographic assets stored in the remote node and, if necessary, transmits the results to the user. Note that Pi and Di are also created by a cryptographic tunnel created with the same mechanism as Ui and Pi: this part of the protocol has been removed from Figure 1 for simplicity.

Figure A-2 shows a simplified version of the Di circuit. The system consists of two microcontroller units (MCUs), one dedicated to (external) communication and the other dedicated to cryptographic operations. This architecture is insulated to protect part of the connection from part of the cryptographic operation using a protected asset.

Figure B-2 and Figure C-2 show this insulation. On the left, in Figure B-2, the box shows the modules that are active during communication with the outside world. During regular operation, this part is active, and the second part, the part dedicated to calculating cryptographic operations, is unchanged. Relays are used to implement this insulation. When the cryptographic module is in operation, as shown in Figure C-2, the part that communicates with the rest of the world is powered by the power supply. The secure cryptographic module runs on an internal memory that is not directly accessible from other routines.



(C) During active cryptographic operations (b) During active communication (a) Public system

Figure 2: Block diagram of Di components highlighting the active part during communication and during cryptographic operations.

The two modules are insulated using a relay. Initially, two relays are in the default state. The microcontroller module that communicates with the outside world is powered by a battery, while the microcontroller dedicated to cryptographic operations is quite clear. When data is prepared in an unstable state, the data is ready in memory (NVRAM), in which case it is executed using magnetic memory (FRAM) and cryptographic operations need to be performed securely. The internal module is on and the external module is off. The safe module draws power from a capacitor to isolate it from the public power supply. The only common components (NVRAM) used to exchange data with a cryptographic engine. However, the power supply (NVRAM) is connected to one source when operating in one mode and to another in another case. (NVRAM) has a simple serial interface designed to ensure isolation between the outside world and the cryptographic module.

Below, we list the main components of the solution. The battery is the main source of electricity. A microcontroller (called an external MCU) holds the processed data (e.g. cryptographic operations for computing) and is powered by a relay.

Then there is a supercapacitor (with about twelve farads) that is connected to another relay. Another microcontroller (called the internal MCU) performs cryptographic operations using an internal memory that holds the user's core materials. This memory uses the same power supply line (not shown here) with the built-in microcontroller. There is a high-strength ferromagnetic memory (NVRAM) and a solid state relay array (SSRA). A complex programmable logic switch (CPLD) (which is completely passive) takes care of switching all the relays to one side or vice versa, as further explanation, through the control signals. The NVRAM connects to both the internal MCU and the external SSRA (alternately to one or the other) via a serial peripheral interface (SPI). The CPLD connects to both the internal (MCU) and external (MCU) via a Universal Synchronous Transmitter Control (UART) channel (also switched by the SSRA).

Discussion and conclusion

This article describes a new architecture and communications protocol for managing and storing cryptographic assets, including secret keys and hidden wallets. The proposed infrastructure is a distributed system using a layered cryptography communication protocol. The private cryptographic keys that control mobile cryptographic resources are divided into a number of different nodes, which are physically far apart and provide the inherent redundancy of distributed systems.

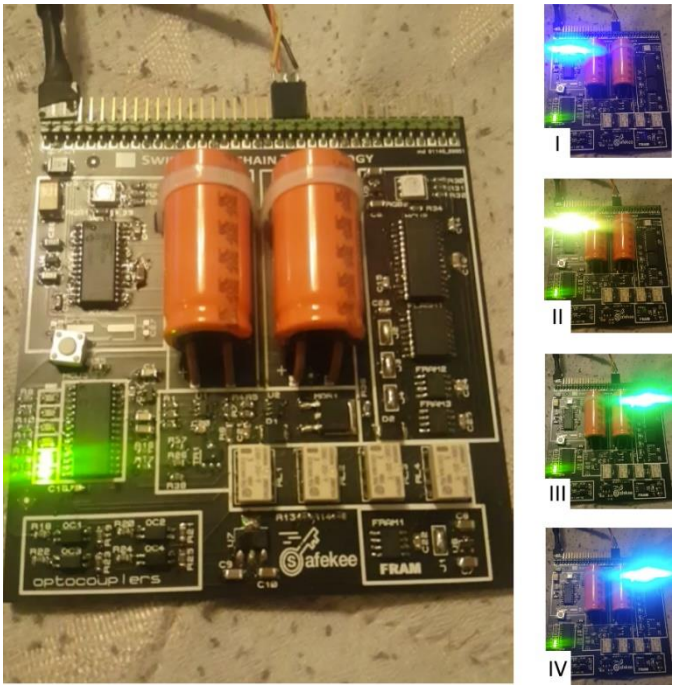


Figure 3. Functional prototype. On the right, the various steps can be seen through the LED indicators:

A) The external communication party is active and the transaction is received.

B) The transaction is stored in NVRAM.

C) The communication side is disabled and the cryptographic side is enabled, transactions are loaded from NVRAM.

D) After signing, the transaction is stored in NVRAM.

Further protection of assets is provided by an isolated electronic architecture that allows secure cryptographic operations to be performed remotely. A dedicated mechanism for isolating hardware cryptographic units on remote nodes is conceived and only works when needed. Imaginary architecture is an extension of the concept of "cold storage". Remote node devices not only store cryptographic material, but also generate private keys internally, so they are never exposed to the outside world.

This system has shifted to physical security more than processing speed. Usually, a digital signature transaction system is able to generate a large number of signatures per second because it communicates directly with the outside world. The system described here is needed in order to be able to separate the cryptographic part, such as loading the capacitor cloud and having switched ferromagnetic memories which must be read and written by an internal and internal processor. For this reason, in a typical usage session that involves charging the capacitor cloud, managing previously signed transactions, and receiving, depositing, signing, and then writing new trades, it is possible to sign several hundred trades per minute.

The performance features of the proposed solution overcome the limitations of many current technologies, and in particular allow end users to maintain direct operational control over their private keys. Because the control protocol used for instructions is extracted from any particular message type, the infrastructure can also be extended to support future applications that require standard cryptographic functions, including solutions beyond cryptocurrencies. We believe that the proposed approach is cost-effective and practical, and is a fundamental step towards a new generation of secure devices that enable remote storage and management.

References

1. S. T. Ali, D. Clarke, And P. Mc Corry. (2015). Bitcoin: Perils of an unregulated global p2p currency. in Cambridge International Workshop on Security Protocols. Springer. pp. 283–293.
2. C. Mann And D, LoebenBerger. (2017). Two-factor authentication for the bitcoin protocol. Int. J. Inf. Sec. vol. 16, no. 2, pp. 213–226

3. Q. Wu, X. Zhou, B. Qin, J. Hu, J. Liu, and Y. Ding. (2017). Secure joint bitcoin trading with partially blind fuzzy signatures. Soft Compute. vol. 21. no.11.pp. 3123–3134
4. M. Gentilal.P. Martins, and L. Sousa. (2017). Trust zone-backed bitcoin wallet. in Proceedings of CS2@HiPEAC. (2017). pp.25–28
5. M. Crosby, *Pattanayak.* Verma, and V. (2016). Blockchain technology: Beyond bitcoin. Applied Innovation. vol. 2, pp. 6–10
6. Ledger. https://www.ledgerwallet.com/, accessed: 2018-03-1
7. Casa. https://keys.casa, accessed: 2018-03-1
8. Bitgo. https://www.bitgo.com/, accessed: 2018-03-1
9. Trezor. https://trezor.io/, accessed: 2018-03-1
10. J. P. Conley et al.(2017). Blockchain and the economics of crypto tokens and initial coin offerings. Vanderbilt University Department of Economics. Tech. Rep
11. R. Gennaro, Goldfeder and A. Narayanan. (2016). Threshold optimal DSA/ECDSA signatures and an application to bitcoin wallet security. in 14th ACNS .(2016). pp. 156–174
12. S. Mavrovouniotis and M. Ganley. (2014). Hardware security modules. in Secure Smart Embedded Devices. Platforms and Applications. (2014). pp. 383–405
13. E. Hofmann, U. M. Strewe, and N. Bosia. (2018). Concept—Where Are the Opportunities of Blockchain-Driven Supply Chain Finance. Cham: Springer International Publishing. (2018). pp. 51–75
14. P. Chen, Jiang, and C. Wang. (2017). Blockchain-based payment collection supervision system using pervasive bitcoin digital wallet. in WiMob. (2017). pp. 139–146
15. T. Bamert, Decker, R. Wattenhofer, and S. Welten, (2014). Bluewallet: The secure bitcoin wallet. in STM. (2014). pp. 65–80
16. N. Zeilemaker, Z. Erkin, P. Palmieri, and J. A. Pouwelse. (2013). Building a privacy-preserving semantic overlay for peer-to-peer networks. in WIFS (2013). IEEE. pp. 79–84
17. P. Palmieri and J. A. Pouwelse.(2014). Key management for onion routing in a true peer to peer setting. in IWSEC. (2014) vol. 8639, Springer.pp. 62–71
18. T. Güneysu, F. Regazzoni, P. Sasdrich, and M. Wójcik. (2014). THOR - the hardware onion router. in FPL. (2014). pp. 1–4
19. D. J. Bernstein, "Curve25519: New Diffie-Hellman speed records.(2006). in PKC. (2006). pp. 207–228
20. The salsa20 family of stream ciphers.(2008). in New Stream Cipher Designs - The ESTREAM Finalists. (2008) pp. 84–97
21. P. F. Syverson, D. M. Goldschlag, and M. G. Reed. (1997). Anonymous connections and onion routing. in S&P 199. pp. 44–54.