NUST Journal of Engineering Sciences, Vol 9, No 2, 2016, pp 60-67

A Smart Card Based Security Extension for The Bitcoin Wallets

Majid Amjad Hussain, Sadia Khalil and Shahzad Saleem
National University of Science and Technology
School of Electrical Engineering and Computer Science
Islamabad, 44000
majidamjad.h@gmail.com, shahzad.saleem@seecs.edu.pk
Submitted: Accepted:

Abstract

With the increase in the use of virtual currencies across the globe, the security of Bitcoin wallets has become a serious concern for the Bitcoin community. The developers are trying to implement concrete security solutions in Bitcoin wallets to ensure that no vulnerability gets exploited. However, a large number of known, as well as zero-day attacks, are launched on the Bitcoin wallets on a daily basis, resulting in a loss of bitcoins. In this regard, this paper presents a security analysis of existing Android wallets. We demonstrate how the implemented security practices can be bypassed by malicious entities, causing financial loss to Bitcoin users. As a countermeasure, we present a smart card based authentication scheme which will protect the users from all of the identified attacks.
Keywords: Bitcoin Wallet, Smart Card, Authentication, Encryption

Introduction

Crypto-currencies like Bitcoin have redefined the meaning of financial transactions by providing a virtual platform for creation, circulation and transformation of bitcoins through anonymous and irrevocable transactions. In 2015 more than 100,000 merchants were accepting bitcoin. People are adopting Bitcoin for transfer of money due to its convenience and lesser transaction fees. At the time of writing Bitcoin's market capitalization is $5.7 billion (Crypto-Currency Market Capitalizations, 2016). However, with this increase in market acceptance, Bitcoin has to face a lot of security challenges.

Bitcoin wallets are very important entities in the Bitcoin architecture. Just like physical wallets, they are used for storing cash and are responsible for keeping a record of all transactions. The security of a Bitcoin wallet is critical to the whole Bitcoin architecture as it is responsible for creating Bitcoin addresses as well as for storing private keys associated with the addresses. If the private key gets compromised, all of the bitcoins associated with that Bitcoin address get lost. Therefore, the major focus of most of the developers is to ensure that private keys in the Bitcoin wallets are secured.

A strong encryption scheme is a prime solution for preventing the wallets from attacks by malicious entities. However, the attackers have still found ways to bypass such schemes. If an adversary gets hold of the phone, he/she can bypass authentication and then can carry out transactions on user's behalf. A number of schemes have been proposed for the protection of Bitcoin wallets, each having its merits and demerits.

Nowadays, attackers are writing malware which can extract and send private keys from the local storage of a user's device. In the case of device theft, an attacker can simply carry out a transaction, as there is no security mechanism implemented. Keys can also be stored offline, such as in a USB. This provides security but at the same time creates usability and availability issues. The offline media should be available all the time in order to carry out transactions. Password protected (encrypted) wallets provide much more security than any other wallets. A key is derived from the password chosen by the user and then the whole wallet is encrypted with that key. The security of the wallet is dependent on the key, which is dependent on the password chosen by the user. Normally users don't use best password practices and choose passwords which have a very low entropy. This motivates an attacker to perform brute force or dictionary attacks. Another threat to wallets is spyware or keyloggers which can log the keys and send it to the attacker. The attacker then generates the key with the help of the password. Sometimes keys for bitcoin transaction are derived with the password, which can be compromised by the above-discussed methods. Hosted wallets are the wallet services provided by third party vendor via a web application. In these types of wallets, the user does not need to manage or secure the private key as key management is handled at service provider's end. In this case, the user is dependent on the service provider's security.

Literature Review

Shayan Eskandari et al. (Shayan Eskandari, 2015) propose a scheme in which the keys can be stored in

the phone's local storage which can be accessed by the application at any time in order to carry out the transactions. Despite this offline/local storage benefit, any other application that has access to the physical storage can steal the private keys. Such scenario is prone to malware attack in which malware can extract and send private keys to any malicious entity.

Angelica Montanez (Montanez, 2014) examines different bitcoin wallets with respect to forensics aspect. The paper discusses different techniques that can be used to get useful information about any installed Bitcoin wallet. This information can be the date of installation, updating or deletion. Log files can be manipulated to find transaction information, IP addresses of peers as well as a number of bitcoins in the wallets. During the examination of Litecoin wallet application, it is found that .93 file contains the private key as well as date and time stamp. Any entity who has access to this file can steal the money easily. This file is accessible to everyone after rooting the device. With the help of public key found in the log file, one can search the address the public ledger to check all previous transactions to connect to address.

Christopher Mann et al. (Loebenberger, 2014) present a thresh hold scheme as a solution to device theft in case of unencrypted wallets installed on it. In order for a transaction to take place, more than one signature are required. To prevent a single point of failure, n out of m signature is needed to proceed with a transaction. This resolves the problem of theft in a way that in the case of theft of one device, then the attacker cannot spend bitcoins. However, this solution causes a problem of increased transactions size which ultimately leads to increase in transaction fees. Another solution provided is of splitting the key to two devices, one mobile phone, and another desktop. The user needs to use a desktop application to initiate the transaction; then it generates a QR-code to be scanned by the mobile application for the purpose of authenticity. A secure TLS connection is built, and part of the key is exchanged to proceed with the transaction. This solution takes less internet and file system usage, as well as less transaction fee as transaction size it small. The drawback of this solution is the availability of both devices is required at one time.

Steven Goldfeder et al. (Steven Goldfeder, 2014) provide a mechanism to secure bitcoin wallet via threshold digital signatures. The technique is based on a threshold value 't' which is actually predefined in the system. First, we have to provide a private signing key to each entity in the system. If we want to carry out a bitcoin transaction, we need the digital signatures of at least 't' entities in order to carry out the transaction. The mechanism implements the 'separation of privileges' principle where multiple privileges are required to perform a task. This solution is feasible for office environments where the consent of more than one person is required for a transaction to be made. Multiple signatures also create a problem of increased transaction size and transaction fees.

Rostislav Skudnov et al. (Skudnov, 2012) describe different types of bitcoin clients w.r.t. their usability and security. Full clients are those who download the whole blockchain and require excessive storage and more network bandwidth. Mobile based clients need to download block headers and hence require less storage space and the network is required. Bitcoinj is java library used for this purpose. Thin clients are the browser based clients, where no computation is on the client side, most of the tasks are being done on server side. In this type, the user doesn't need to secure the private key. As most of the things are managed on the server side, therefore if the server is compromised then all the system is compromised. Mining clients provide high CPU or GPU based computation in order to run bitcoin system. Some other clients include signing-only clients, deterministic wallets, and brain wallet. The bitcoin release 0.4.0 provides the facility to encrypt the wallet with a passphrase to prevent the wallet from an attacker since 0.4.0 release bitcoin wallets are able to read the encrypted wallet and decrypt it at the run time if correct passphrase is provided. Paper backup is a technique where the user takes the backup of his/her private key on paper instead of digital media. For this purpose, QR-code is being used in order to avoid the typing mistake. The printed paper with the correct QR-code having private key can be accepted as payment. The receiver can send a transaction by scanning the QR-code and signing it with the scanned private key.

Bitcoin Wallets contain public and private crypto keys, other passcodes, and PIN. This makes them prone to a large number of theft attacks. If an adversary gets hold of the phone, he/she can easily bypass authentication and then can carry out transactions on user's behalf.

The authentication process used by the Bitcoin Wallets to make sure that only the authorized person is using the application can be bypassed. The existing authentication scheme stores passcodes and PIN in internal storage (shared preferences as XML file or as SQL database) and believes that the information

stored cannot be accessed by another application not even by the owner of the phone. However, the files can easily be accessed by anyone who can get root access to the device. The existing authentication scheme can be analyzed using the application logic written in code. This can be exploited by simply decompiling the Android .APK file. Some Bitcoin Wallets are storing the hash (MD5 and SHA1) of passcode and PIN which can be easily cracked within seconds with the help of web-based cloud service which already have the computed hash chains (CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc., 2016).

Sometimes application developer relies on a cryptographic solution (encryption, decryption). With this approach, the developer accepts PIN code from the user and generates a key from this PIN code and then encrypts all data with it and stores nothing in the internal memory of the phone, except the encrypted file. At the time of decryption, the application again accepts the PIN, generates the key, and then decrypts the file. If the file is decrypted into a correct format, this means that the provided PIN is correct; otherwise, the PIN is considered incorrect. Normally the length of the PIN is 4 digits, what means an adversary only need 10^4 tries to log into to the system successfully. To overcome this issue, the applications put a limit of 2-5 on the number of unsuccessful tries for the PIN. An attacker can still bypass this mechanism, the attacker simply roots the device, then extract the encrypted file and brute force on his/her system without the limitation of no of tries. He/she can achieve results within few seconds with a simple laptop.

Motivation

Case Study 1: Coinbase Authentication Bypass

Coinbase is an Android based Wallet which contains a design flaw which leads to a severe vulnerability which allows an adversary to bypass PIN authentication. This allows the adversary to open an Android Wallet without knowing the PIN. With this design flaw, the adversary can steal and manipulate user information (e.g. username and email). Apart from that, this allows the adversary to steal all the Bitcoins present in the wallet.

Android applications can store data in either and SQLite3 database or and XML file (called shared preferences). Coinbase Android Wallet makes use of the XML file to store information regarding PIN.

By making certain changes in the menShared_Preferences.XML file we can bypass PIN authentication. By deleting the XML statement used for storing pin related information, an attacker can mislead the application into launching with a home page instead of the application pin authentication page. The home offers the ability to steal Bitcoins from Wallet. The steps used for exploitation are explained below and are shown in the figure no 1.

1. Set PIN for authentication.
2. User logs into the system.
    a. User data is stored in com.coinbase.android_preference.xml
3. Login successful and the home screen is prompted.
4. The attacker edits com.coinbase.android_preferences.XML accordingly and replaces the actual file with the edited one.
5. Attacker opens the app and sees the home page instead of authentication screen.

6. Case Study 2: Bitcoin Wallet Security Bypass

7. 'Bitcoin Wallet' is an Android-based open-source Wallet application, which can be downloaded from the official site of Bitcoin as well as on Android Play Store. On Android Play Store it has 500,000 - 1,000,000 downloads which mean this Wallet is popular among Bitcoin users. For the purpose of safety, this wallet uses a PIN code to put a lock on spending of Bitcoins. Wallet only accepts the numeric value, which means that we can only enter digits in the range zero to nine. Figure 2 shows that the application marks the pin "strong" even when the user enters only 8 digits. This application also implements the tries policy, which means that a user cannot enter a PIN code after a certain number of tries. With this PIN, the application generates a key and encrypts the whole wallet and saves it to internal storage. An adversary can root the device, pull the encrypted file and apply brute force using 10^8 combinations.

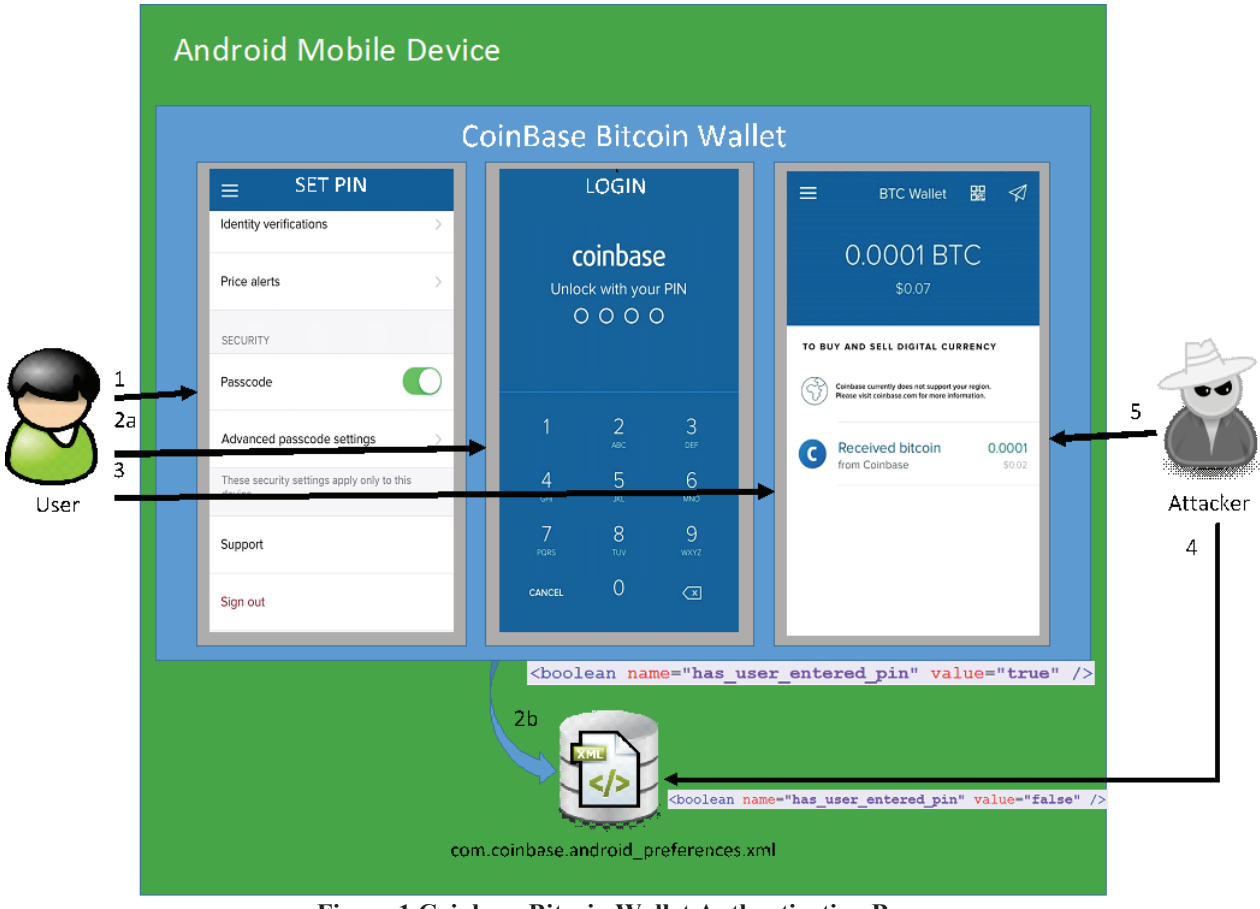

Figure 1 Coinbase Bitcoin Wallet Authentication Bypass

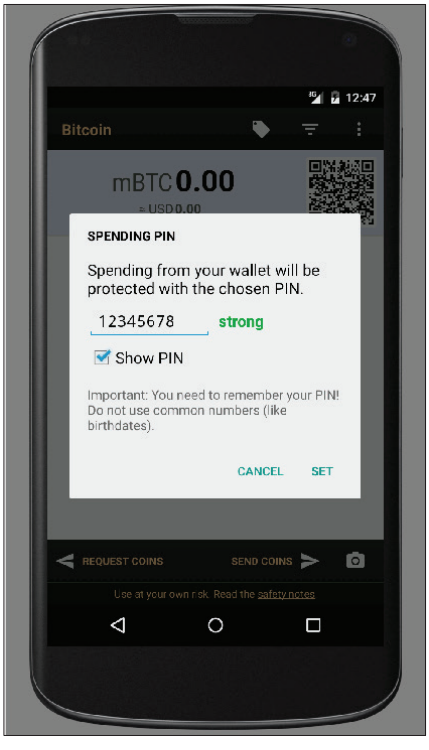

Figure 2 Bitcoin Wallet

Problem Statement

Once an adversary gains physical access to the device running Android based bitcoin wallets, it can bypass the authentication mechanism and can steal Bitcoins by carrying out transactions on the behalf of users'. This creates a need for an efficient and secure solution which can ensure the security of the wallet security even if the device is not in possession of the user.

Proposed Solution

We have noted that the Android device is not safe for storing users' credentials and important keys. So we are proposing which involves keeping the keys and other necessary data out of the Android device for the purpose of authentication. For this, we propose a smart card based authentication scheme for Android based wallets. Figure no 3 shows the architecture of scheme proposed for a secure authentication for Android-based Bitcoin Wallets.

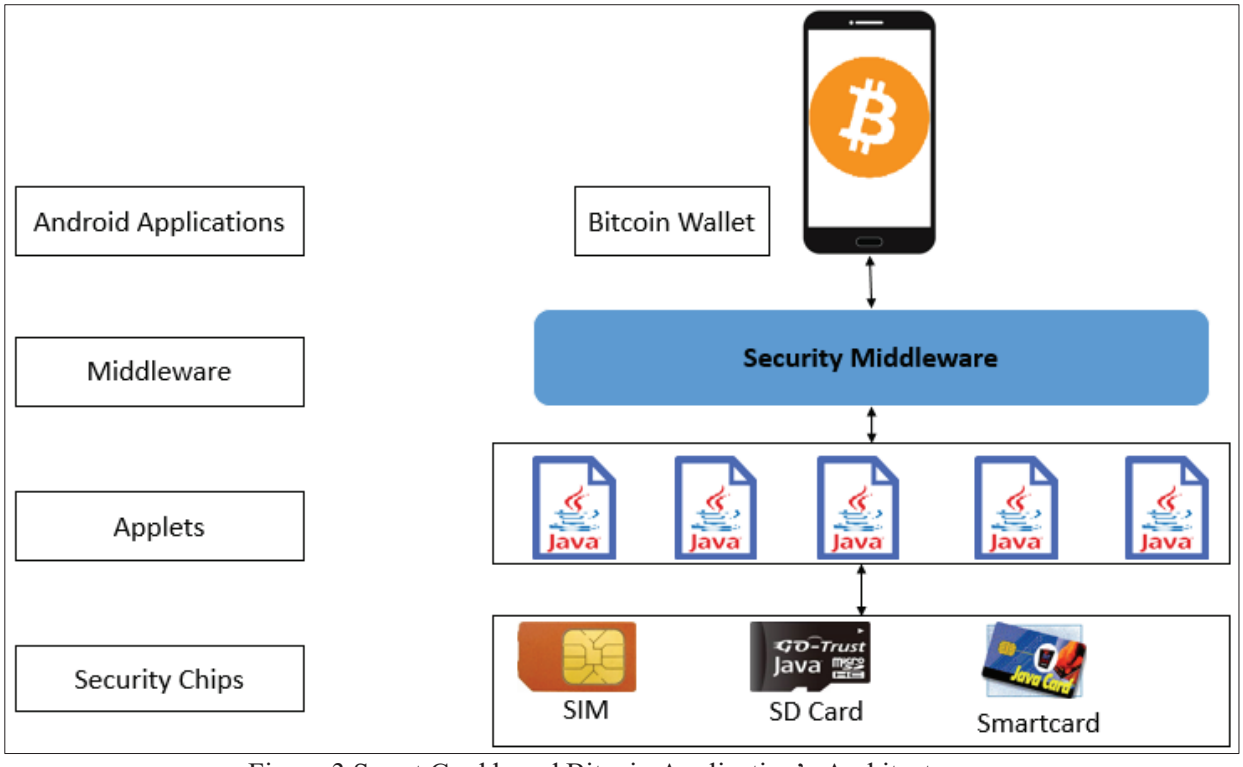

Figure 3 Smart Card based Bitcoin Application's Architecture

Go-Trust Technologies launched microSD secure memory card, which combines advanced features of smart card technology and USB mass storage into a convenient microSD form. This is a very smart solution to secure portable devices. Go-Trust launched the SDK for accessing smart card chip through file-system based interface for different mobile platforms. Its extended benefits include for application security in JavaCard based solution, PKI support (i.e. digital certificates and cryptographic keys storage, strong authentication, digital signature, data encryption for a perfect PKI integration) and standard mass storage facility (i.e. 2 GB to 8 GB) (GO-Trust Encrypter Family, 2016).

There are two types of Bitcoin Wallets: "heavy–weight" – (full application) Wallets and "light–weight" (GUI / communications only) Wallets. Both the types can be used with PC stations as well as smartphones. Two main characteristics of the two types of Wallets are:

1. Full Wallet stores all security credentials and blockchain transactions locally; and has direct connection with the Bitcoin network
2. Lightweight Wallets provide only a GUI-based communication with users and a connection to the Wallet server. The security credentials and blockchain transactions are all stored at the Wallet server. Wallet Server also performs communication with the Bitcoin networks.

For local security extensions, both types of Wallets are equivalent, but the approach is different. In the case of full application wallets, all security extensions are local (at PCs and smartphones), whereas in the case of lightweight Wallets, security extensions are implemented at the server.

In this research, both the types of Wallets will be considered. As a full application, the open–source application Bitcoin Wallet will be used.

Design

Table no 1 shows the issues/challenges and their solution which we are proposing:

Table 1: Issues/challenges and their solutions

Sr.	Issues/challenges	Solution
1	Poor/weak or no Authentication while accessing the Bitcoin Wallet	Smart Card Based Authentication
2	Encryption of Bitcoin Wallet with weak key, which can be derived from Password/PIN	Encrypt wallet with smart card's key
3	Protection of the Backup	Encrypt backup with Smart card's key
4	Prevention from Malware Attacks	Custom keyboard
5	Prevention from Social Engineering Attacks	Random keyboard every time

For the problems discussed above, we propose a smart card. Solution is derived from FIPS 196 (This standard specifies challenge-response protocols by which entities in a computer system may authenticate their identities to one another and a paper "Smart Card Authentication for Mobile Devices" (Jansen, Smart Card Authentication for Mobile Devices, 2016) and its report (Jansen, NISTIR 7206, Smart Cards and Mobile Device Authentication: an Overview and Implementation, 2016), published by NIST.

Poor/Weak or No Authentication

Existing bitcoin applications either use no authentication or use weak/poor authentication. By poor or weak authentication we mean a simple pin code, which can be found easily either in preferences.XML file or in the database file. This pin can also be brute forced easily. To overcome this problem we are proposing a smart card-based authentication.

Steps:

1. The first user downloads the android bitcoin application from android play store. This application has a CA certificate.
2. The user requests the service provider to provide a smart card. The service provider provides the smart card, which has a private key ($\mathbf{K_R}$) as well as a user certificate.
3. The application retrieves the certificate from the smart card and stores it. After retrieving the certificate from the smart card, it first verifies it. As the application has a CA certificate, it matches the CA signature on the certificate provided by the smart card.
4. The user enters the default pin code to authenticate with smart card and sets a new pin code.
5. The initial or registration process ends here. Now it's time for authentication.
6. At first, the user enters pin code, if the pin code is correct the smart card will allow the user to proceed, otherwise it will terminate the authentication process and lock the smart card after three unsuccessful attempts.
7. If the pin code is correct, the application generates a random number "**A**" and sends it to the smart card.
8. The smart card generates a random value "**B**" signs "**A||B**" with the private key on the card ('||' denotes concatenation), and returns "**B**" and the signature to the application.
9. The application retrieves the user certificate and extracts the public key "$\mathbf{K_U}$", verifies it, and then verifies the card's signature over "**A||B**" using the public key contained in the certificate.
10. If everything verifies successfully, the authentication succeeds; otherwise, the authentication attempt fails.

Figure no 4 explains the flow of different activities.

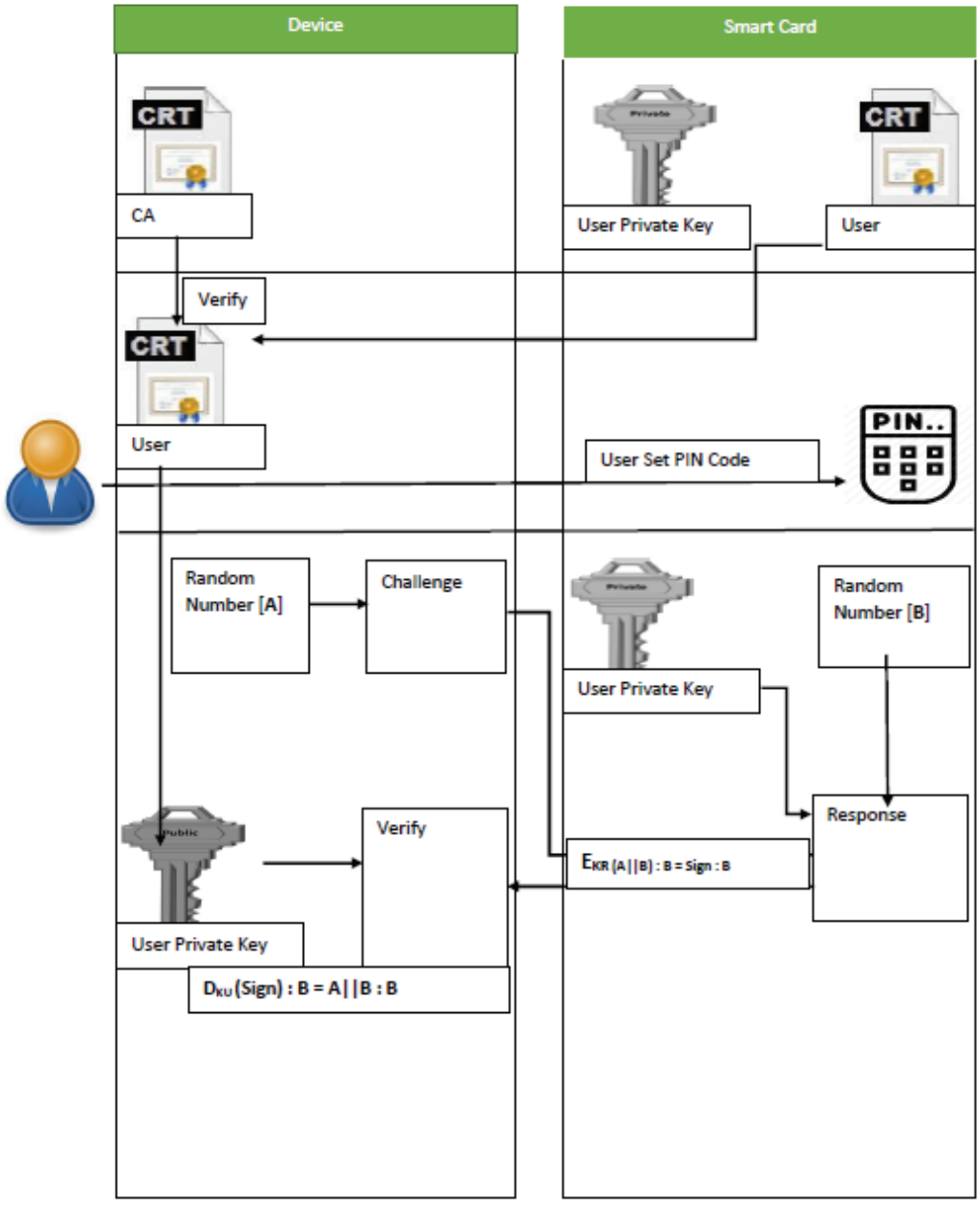

Fig. 4: Smart Card based Authentication

Encryption of Bitcoin Wallet with Weak Key

BTC Wallet provides protection to the wallet by encrypting it. BTC Wallet encrypts the wallet with the PIN provided by the user. It is very easy to brute force only numeric values.

Our proposed solution is that, first, the smart card takes the hash of the user's pin, then encrypts it with the private key stored on the smart card.

$$\text{New Key} = ENC_{KR}\,(\text{HASH (PIN)})$$

The newly generated key is sent back to the application for wallet encryption.

Another solution is that by using a smart card with a good processing power and memory, we could send the wallet file to the smart card and the smart card would encrypt it with its own private key. ENC_{KR} (Wallet File).

Protection of the Backup

BTC Wallet provides the mechanism for backup in the case of a lost phone or if some malware corrupts the Wallet's data or its private payment key. BTC Wallet, at the time of backup, asks for a password to encrypt the Wallet, so that only the true owner can access the data. The wallet is useless for others as it is encrypted. But it is observed that people often don't. As users use low entropy passwords, the attacker takes advantage of this and figure out the passwords easily in most of the cases. To provide enhanced security, we propose the encryption of the Wallet at the time of backup, with the private key of the smart card.

Prevention against Malware Attacks

Android has 82.8% market share of smartphones. Attackers are writing much more malware for Android as compared to that for any other mobile operating system. Nowadays, a number of malware and rootkits are infecting Android devices. Most of the malware install a key logger on the device and try to capture credentials. Keylogger gets the root-level privilege and listens to system calls made by the application and capture keystrokes. The solution to this problem is writing a custom keyboard.

Prevention against Social Engineering Attack

Humans are known to be the weakest link in security. For example, imagine a person sitting in front of you, observing you while you enter the PIN into to BTC Wallet application. By carefully observing the movement of fingers, he/she can detect the correct pin. To avoid this, we propose a custom keyboard which shows random keys every time.

Implementation

We took an open source android bitcoin application (bitcoin-wallet, 2016) and implemented our security extensions according to the given architecture. Smart card and android emulators were used to test the whole scenario.

Conclusion

Phone storage is not a secure place to store important credentials, especially in the case of rooted phones. The use of a smart card provides much more security.

REFERENCES

1. Bitcoin, 2016.
2. bitcoin-wallet, 2016.
3. "Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc," 2016.
4. "Crypto-Currency Market Capitalizations," 2016.
5. G.-T.E. Family, 2016.
6. W. Jansen, et al., "Smart Cards and Mobile Device Authentication: An Overview and Implementation," *NIST, NISTIR*, vol. 7206, 2005.
7. W. Jansen, " Smart Card Authentication for Mobile Devices," 2016.
8. C. Mann and D. Loebenberger, "Realizing two-factor authentication for the Bitcoin protocol," *IACR Cryptology ePrint Archive*, vol. 2014, 2014, pp. 629.
9. A. Montanez, "Investigation of Cryptocurrency Wallets on iOS and Android Mobile Devices for Potential Forensic Artifacts," 2014.
10. S. Eskandari, et al., "A first look at the usability of bitcoin key management," *Proc. Workshop on Usable Security (USEC)*, 2015.
11. R. Skudnov, "Bitcoin clients," 2012.
12. S. Goldfeder, et al., "Securing Bitcoin wallets via threshold signatures," 2014.