Proof of Concept Key Escrow over NFC for use in Blockchain and Self-Sovereign Identity

Robert Baart,
Tim Bruyn,
Gouri Calamur Viravalli,
Bartosz Czasyński,
Dorian de Koning

February 2, 2018

Abstract

Public-private key pairs lay at the very heart of access control to many services available online. Loss of private key usually leads to loss of funds or online identity which cannot be restored. Key escrow is a way to reconstruct this private key with assistance of trusted third parties. This paper presents an Android implementation of a key recovery scheme based on Shamir's secret sharing, where key shares are transported over Near Field Communication (NFC). The owner gains the ability to reconstruct the key with the help of trusted parties who hold preshared pieces of the key.

1 Introduction

The blockchain was created by combining existing technologies which are proven to be secure, namely the Internet (as an underlying Peer-to-Peer network), public key cryptography, and protocols governing incentivization. It is a distributed and decentralized computation network that provides participating users with a robust way of verifying asset flow, thanks to public transaction recording. User identity and their ability to exert control over the assets that are bound to their identity is of crucial importance for correct functioning of the system. Authentication and authorization are two processes that are enabled by application of public key cryptography - users are given an option to decide on who they would like to share information with, while also providing an ability to prove who created the message. Authorization requires a distributed network and this reduces the risk of a single point of failure or centralized corruption. Although there might be problems revolving around this new concept of public ledgers, these arise mostly from human error or entities with bad intentions, and not necessarily due to flaws in the concept.

One of the main issues concerning blockchain technology is privacy. Storing data on a public ledger requires publishing all information about transactions and participating entities onto a public database. It can be read without any access restriction, by the public and governments.

While blockchain removes the central point of failure for the data contained in it, each participant in it has become its own central point of failure for maintaining their identity, the private key they use for communication, on the blockchain. The very decentralized nature of the blockchain makes these identities unrecoverable when lost, hence the need for loss preventive utilities. Key escrow, securely lending fragments of the private key to trusted parties so it can be recovered after a loss, is a candidate for such a loss prevention system.

2 Problem Statement

The most important piece of information, governing access to all the information a user has claims to, is his private part of the key pair he owns. Within the identity management environment this means being able to prove one's identity and being allowed use of services, ranging from administrative to financial. In the world of cryptocurrencies private keys empower anyone who holds it to issue transactions and claim ownership over any commodity assigned to the key pair.

Due to very sensitive nature of this information, special care has to be taken in order to ensure its security. Lost keys render anything assigned to them nontransferable, effectively freezing funds, or ruining ownership claims.

In order to ensure desired functionality of a blockchain-based system allowing for transactions between various parties, a thresholded sharing scheme is proposed. The scheme aims to provide a key recovery functionality, given existence of a number of trusted third parties with whom a user is willing to share parts of his secret. Upon losing access to the private key, the user will be able to contact the trusted third parties and obtain from each of them a partial representation of his secret. Given that the trusted parties are indeed trusted, and will not collude with each other in order to obtain the full secret, no information about it can be derived from the partial secret representations.

3 Concepts used

3.1 Key escrow and recovery

A key escrow system is one wherein cryptographic keys are stored by an authorized trusted third party. The keys are split up and distributed among different parties such that a given number of k parties, where $k > 1$, must cooperate in order to retrieve the secret key. Key recovery can be used for backup and restoring of lost cryptographic keys, as well as be a basis for threshold encryption. Key escrow is a method of involving a third party and granting it access to (parts of) the key, hence authorizing data access or acting on one's behalf. Accomplishing key recovery using the idea of key escrow helps serve as a backup mechanism as it ensures that the owner can re-create his/her private key and regain access to encrypted data, in the event that the key is lost.

3.1.1 Shamir's Secret Sharing

The specific method of key escrow used is Shamir's Secret Sharing [1]. This scheme is based on polynomial interpolation; only one polynomial of $k-1$ fits k points in a plane, but any number n of such points can be distributed. The secret s is the constant of the polynomial. Modular arithmetic in a finite field G of integers up to some prime p is used, which makes the polynomial less predictable for attackers with access to some of the shares. This means the prime should be larger than the secret and than n, effectively becoming a security parameter.

To construct key shares, a random polynomial of degree $k-1$ is created with the secret as its constant, and a suitably large prime p is chosen. For n numbers x_i chosen uniformly randomly from the numbers up to p the value y_i of the polynomial is calculated, and the tuples (x_i, y_i) are the shares to be distributed. With k shares, the polynomial can be reconstructed easily, with $k-1$ shares there are p possible polynomials.

1. **Secret share generation:**
   - Choose coefficients $f_i, ..., f_{k-1} \in G/pG$
   - Let $f(x) = s + f_1 * x + ... + f_{k-1} * x^{k-1}$.
   - Distribute $f(i)$ to a party i, $i = 1, ..., n$.
2. **Secret recovery:** Lagrange Interpolation method over finite field can be used to define a system of linear equations.

$$f(z) = \sum_{i \in G} x_i \prod_{j \in G, j \neq i} \frac{z-j}{i-j} \tag{1}$$

Through use of Gaussian Elimination the equation can be simplified to:

$$x = \sum_{i \in G} x_i \prod_{j \in G, j \neq i} \frac{-j}{i-j} \tag{2}$$

The product element in the right-hand side of the 2 is a constant independent of x, which allows it to be precomputed in order to reduce secret recovery to a linear time, ensuring efficiency. [2]

Shamir's secret sharing has the property of information theoretic security. An adversary with unlimited computational power does not have enough information to be able to break the cryptographic scheme. Even with $k-1$ shares there are p polynomials to guess among. This can be represented as an underdetermined system of polynomial equations. Without having access to at least k secret shares, too many degrees of freedom are present in the system and any value is a consistent solution. The problem becomes as hard as directly attempting to guess the secret, given that polynomial coefficients are drawn from a uniform distribution. This property was the main factor that contributed to the choice of Shamir's Secret Sharing for the purpose of threshold secret sharing. Distribution of key among independent trusted third parties positively impacts the security of the system, requiring multiple points of compromise in order to successfully retrieve the secret. Besides recoverability, the scheme also offers renewability property for the shared secrets. In case any of the trusted parties loses his share, or an additional trusted party joins, a new secret share can be easily issued. This should be done with utmost care, as generation of additional points on the polynomial can weaken the security of the scheme.

3.2 Threshold Cryptosystem

Threshold encryption cryptosystem is one in which a number of parties are involved, and a minimum of the threshold number of parties are required to perform the decryption operation to successfully retrieve the message. In an asymmetric setting a public key is used to encrypt a message, and the private key is divided and distributed among different parties. An inverse protocol can be used to produce a threshold signature, where multiple parties have to cooperate to produce a valid signature, which is publicly verifiable. Authors of [3] have proposed cryptographic schemes allowing for cooperative encryption and decryption of messages with possibility to extend them to thresholded signature schemes. Threshold encryption functionality would add additional security layer to our current solution, as the implemented system allows only for recovery of a single secret and every new secret requires running of the whole secret sharing protocol again. For purpose of arbitrary message encryption (or digital signing) and no need for a full key recovery during every operation, a fully fledged threshold encryption scheme should be implemented, this however is not necessary for the purpose of recovering a key discussed in this report.

4 Methodology and Deployment

The application described in this paper consists of two components: An implementation of Shamir's secret sharing and a component handling the NFC communication, including the user interface.

4.1 The implementation of Shamir's secret sharing

For our implementation of Shamir's secret sharing, the GitHub repository 'Shamir' from Coda Hale was used [4]. Shamir's secret sharing algorithm consists of a Java implementation of a Galois finite field and a scheme to encode each byte of the secret as a separate polynomial of degree 8 over the Galois finite field.

4.2 The Android application

The Android application consists of multiple android Activities each having their own responsibilities, an image showing the data flow within the application can be seen in figure 1.
When opening the application for the first time the user is asked to enter his name. When the user wishes to share a new secret he presses a button on the main page of the application after which he is directed to a page for creating shares. On this page a title for this escrow instance, the secret, n and k can be specified. When a "Create shares" button is pressed the information is passed to the Shamir secret sharing library where shares are created. The entered amount of shares is stored in persistent storage and the user is directed to a page indicating how many shares he has to hand out, the method used for passing shares is discussed below in more detail.
If in the future the user wishes to recover a secret he presses the corresponding button to recover a secret in the list of his own secrets. The user is directed to a page indicating how many shares have to be received from friends. Friends can pass shares back to the user by using NFC again. When enough shares are received the secret is recovered and displayed on the screen.

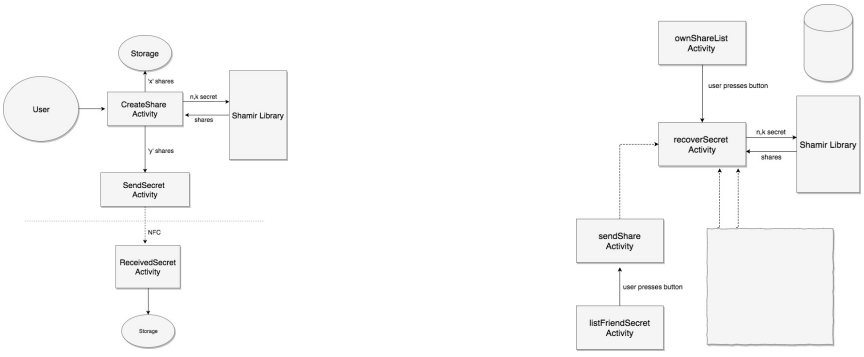
4.2.1 Storage of shares

Information about the shared secrets and the received shares should be available after an app or even device restart, therefore it is stored in persistent storage on the device. Android operating system ensures that no other application can access this stored data.

4.2.2 The NFC communication

Near Field Communication, or NFC, is a protocol that allows two electronic devices to establish a one-way or a two-way connection when they are placed close to each other. The most common use of this concept is in contactless payment systems, and is also used to share data files such as photos, videos, etc. Our implementation uses NFC channel to establish a connection between two involved devices to share and retrieve keys. This choice has been made based on user-friendliness of such an approach and additional security offered by appropriate utilization of NFC standard. Compared to showing the share in text form or transferring it via scanning a QR code it has some advantages. It allows for much larger shares and thus secrets, mistakes in copying do not occur due to the protocol, and no camera or passerby can see the share. As NFC relies on a wireless connectivity to transfer data, it does inherently introduce risks associated with the communication medium, which have to be handled in an appropriate manner. Secure channel should always be enabled between two communicating devices.

- **Eavesdropping -** any identity capable of listening in on wireless communication happening on frequency of 13.56 MHz can intercept the signal and record ongoing conversations. This risk factor is mitigated by two methods. Firstly, the default reception range of NFC is within a distance of less than 1.6 inches, which reduces the chance of message interception, unless a powerful enough antenna or a signal amplifier is used. However, researchers have proven that relying just on the range itself is not a good security measure, as the range of NFC can be easily extended [5]. Second security measure against eavesdropping is the establishment of secure channels between communicating devices, which allows them to encrypt the data in transit and grant access only to authorized devices.
- **Data Integrity -** since all the communication is happening on a detectable channel, additional measures have to be taken in order to ensure no manipulation happened while data was in transit. NFC technology makes use of Cipher-based Message Authentication Codes (CMAC) and Digital Signatures to ensure that no data tampering took place while it was in transit.



(a) Graph presenting data flow through the system

(b) Graph presenting data flow through the system during recovery

Figure 1: Diagram representing data-flow through the application

4.2.3 The user interface

For the user interface of the application a simple vertical layout with the elements following the widely used material design guidelines was chosen.

4.2.4 Communication with external applications

The developed application can be used as a standalone application to share any secret, however to demonstrate the use case an integration with the TrustChain [6] application was made. This integration allows opening the standalone key sharing application and automatically filling in the TrustChain private key.

5 Discussion

A drawback coming from use of Shamir's Secret Sharing scheme is relates directly to the information carried by each of the shares, communication and storage requirements. With access to $k-1$ shares, any scheme participant should not be able to deduct any meaningful information about the secret from the share he is given. This implies that the last remaining share must carry as much information as the secret itself, which in turn requires each of the shares to be of the same length as the secret message. Hence, the transmission and storage requirements for the network are equivalent to the length of the secret and number of shares distributed within.

A disadvantage of any key escrow system is that once shares are distributed, they can in perpetuity be used to recover the private key. There is no mechanism to exclude a friend turned enemy, except by destroying $n-k+1$ shares, which is difficult and impossible to verify. Secrecy of the scheme relies on the assumption that trusted third parties will not collude.

5.1 Alternate uses of key escrow

The most obvious use of key escrow is to store a user's private key in case it is lost. The secret shared does not have to be a key though, it can be any type of information, provided it can be stored in a numeric form and is not too large. Even large secrets can be effectively put in escrow by storing the secret encrypted in some reliable place of common knowledge and distributing shares of the encryption key.

In the event that a user requires a higher level of security, they could consider nested key distribution, wherein after the key is split into shares individual shares are again split using Shamir's algorithm and distributed among different people. This allows for more tailored handling of groups or locations or other sets within the set of people the the shares are distributed to. It allows for example deciding that any two family members plus any three friends can recover the key, but no combination with fewer of either.

Under the assumption of some place where common knowledge can be stored, a user can announce (sign) a successor key and distribute shares for the private part of it to others. While this will not allow them to recover the user's own key, they can prove with the announcement that they hold the successor key. This can be used for example in the case of the death of the user. The user might not want to give full access to their accounts, but does want to give proof of inheritance and access to memorialising online accounts.

5.2 Social groups

The applied schema for sharing the secret is based on the notion of a group being a more trusted third party than a single person. The fact that some shareholders may over time become malicious and might try to collude to recover the shared secret should not however be seen as entirely independent events, groups may turn malicious or lose data together. To avoid this scenario some precautions can be taken, for example shares could be distributed among different social groups that are likely not to know each other.

5.3 Integration with blockchains and use in self-sovereign identity

Key escrow does not solve all the world's problems. A private key can be stolen, either directly or through accumulation of k shares. For the application of sovereign identity, a single private key's security will become less and less as more key escrows are created as a person moves through life and gains and loses friends. A subsection of these problems can be solved by using a sequence of private keys as a person's identity, where each next public key must be approved (signed) by the previous private key and posted on the blockchain.

The key recovered with key escrow could be one of these next keys, which could be used to invalidate the previous key, if for example it has been stolen. Because the blockchain is common knowledge and unforgeable, timers could be used for freezing an identity when such key recovery takes place, allowing the owner to cancel it with the current private key. The parameters of such a system however depend completely on the needs of the self-sovereign identity system one wishes to set up, so this extension should be approached starting from the system to implement.

6 Conclusion

This report has seen how the owner of a private key can create shares of his arbitrary-length key with the help of Shamir's secret sharing algorithm which makes use of polynomials and Lagrange's interpolation, and distribute it to his friends or other such trusted parties in such a way that with a given number of these parties, he can reconstruct his key if it is lost. The keys are shared with the help of a mobile application by making use of Near Field Communication protocols. This makes it convenient for the owner to be able to recover his key quickly, and only in the event that his trustees are physically next to him, hence confirming the identity of the trustees. This application can be downloaded and extended for other uses too, and not just to reconstruct a private key. For example, this application can be used for storing highly sensitive passwords or even bank details of people owning joint accounts. The requirement of the physical presence of all trusted members could pose a disadvantage, but in a world where security breaches are increasing at an exponential level, this could prove to be advantageous.

References

[1] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[2] David Wagner. CS276 Cryptography - Lecture 4.14.04. 2004.

[3] Jonathan Katz and Moti Yung. Threshold cryptosystems based on factoring. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 192–205. Springer, 2002.

[4] A java implementation of shamir's secret sharing algorithm over gf(256). https://github.com/codahale/shamir.

[5] Henning Siitonen Kortvedt and Stig Mjølsnes. Eavesdropping near field communication. 01 2018.

[6] Trustchain. https://github.com/wkmeijer/CS4160-trustchain-android.