A Private Key Recovery Scheme Using Partial Knowledge

Har Preet Singh
FIWARE Foundation
Berlin, Germany
harpreet.singh@fiware.org

Kyriakos Stefanidis
Fraunhofer FOKUS
Berlin, Germany
kyriakos.stefanidis@fokus.fraunhofer.de

Fabian Kirstein
Fraunhofer FOKUS
Weizenbaum Institute
Berlin, Germany
fabian.kirstein@fokus.fraunhofer.de

***Abstract*—In this paper we explore the problem of secure handling of private keys in blockchain applications. We present a novel approach, named "Partial Knowledge Recovery Scheme" (PKRS), which allows for the recovery of an encrypted private key through the use of personal security questions.**

In PKRS, an individual is asked a set of questions, and the answers to those questions are used to encrypt the input and produce a secured private key. Through the use of Shamir's secret sharing algorithm, the original private key can be recovered if the individual can answer correctly only a subset of the original questions. PKRS does not require any external services for the recovery process, since all the required information is stored within the secured private key itself.

This approach tries to achieve a middle ground between security and usability. Security, where the private key needs to be encrypted and safely stored offline. Usability, where an individual wants to be able to recover their private key without the need of an easily forgotten passphrase and be able to store it in their personal cloud environments.

We also discuss the correct design of personal security questions in social environments where an individual's personal data can be mined through public records and social networks. Finally, we present a blockchain Self-sovereign Identity use case, which was used for the integration and evaluation of PKRS within a real-world application.

***Index Terms*—private key recovery, usability, blockchain, identity management**

I. Introduction

One of the main characteristics of blockchain-based distributed ledgers is that transactions are irreversible. Once a validly-signed transaction is on the blockchain, it cannot be removed. Since such transactions usually represent the transfer of digital assets or the execution of a smart contract, the loss of control of the private key can be disastrous. For instance, a stolen private key of a Bitcoin address can lead to significant loss of funds. Since an attacker can easily sign fraudulent and irreversible transactions, it is critical to secure the private key and avoid any misuse.

Naive solutions that users tend to adopt include the use of a centralized cloud service. This is a convenient yet not very secure approach since it depends on the security and confidentiality of said services.

This work has been partially funded by the Federal Ministry of Education and Research of Germany (BMBF) under grant no. 16DII111 ("Deutsches Internet-Institut").

A more secure approach is, of course, to encrypt the private key using a high entropy passphrase and store it on a local, private machine or a specialized security module. The success of this approach depends on the user's capabilities to manage encryption tools correctly, to keep the passphrase memorized, and to ensure the long-lasting operation of the hardware. Even specialized hardware modules can get corrupted taking the stored private key with them.

We argue that, in certain use cases, it would be ideal to enjoy the usability and redundancy of storing the private key in less secure places, such as the cloud services, while avoiding the risk of losing access to it forever due to a bad case of forgetfulness.

In this paper, we present a novel approach for securing a private key based on a combination of Shamir's Secret Sharing (SSS) algorithm [1] and personal security questions, that strives to achieve the aforementioned middle ground. We named our approach the Partial Knowledge Recovery Scheme (PKRS). We also present the implementation and evaluation of PKRS as part of a larger blockchain Self-sovereign Identity (SSI) use case.

The rest of the paper is organized as follows: Section II presents related work on similar schemes, on the usage of security questions as a form of password reset mechanism and some related hard problems. Section III presents our approach and discusses the design of personal security questions. Our use case is described in section IV, and we conclude the paper in section V.

II. Related Work

Shamir's Secret Sharing [1] is a simple yet efficient and secure algorithm that allows to split any type of information, usually a secret, into arbitrary parts and be able to recover it while only having access to a subset of the initial parts. It is apparent that such a method can be very useful to a multitude of diverse kinds of problems. In this section, we briefly investigate solutions that are based on this method and focus on existing approaches that specifically deal with the problem of cryptographic key recovery.

The use of Shamir's secret sharing to the manipulation of cryptographic keys is not something new and has already been used in [2] to split and rejoin PGP desktop keys. In various fields, researchers and companies are leveraging this algorithm

to provide decentralized sharing for their own solutions or products.

Vault [3] created a crypto-storage platform that leverages decentralized storage and decentralized cryptography to provide storage models specifically for owners of crypto-assets. Their approach harnesses the user's trusted circles as a mesh network of mobile devices for distributed storage and, at the same time, as a method of social verification of each owner's identity.

Regarding the problem of searchable encryption, the "serial interpolation filter" [4] has been developed as a method for storing and interacting with datasets, without exposing the original data.

For the general problem of authorization, Ahmadvand et al. [5] presented a generic architecture based on secret sharing, aimed to address critical authorization operations. They also provided benchmarks of different such schemes and analyzed their trade-offs in security, functionality and performance.

Even in the field of electronic voting, given that there is a growing interest in leveraging blockchain technology to provide secure voting solutions, secret sharing can be used to enable on-chain votes submission, and winning candidate determination. [6]

The aforementioned schemes show that the method of secret sharing with partial information can be a valuable tool for a wide range of problems. In the specific problem of cryptographic key recovery, the only solution that is similar to ours is the approach taken by Grid+ [7], which is still, to the day of writing, in its early stages of production. According to this solution, it is possible to recover any key without the need of physical backup by utilizing multiple external actors that hold parts of the key. The actors in that solution are devices, such as their proprietary agent or the user's mobile device as well as an external server. Grid+ solution, while trying to solve the same problem, is limited to the existence of specific purpose devices and services. Our approach offers a more generic solution which only relies on the fundamental source of information when dealing with passphrases, which is the user's brain.

In the topic of security questions as a form of password or private key reset, it is widely speculated that this method sacrifices security in favor of an alternative way of recovery. In fact, security questions are being used by even the more sensitive services, like web banking, as a form of password **reset** and **not recovery** (for obvious reasons). Since our approach allows the actual recovery of a key, it is important to identify whether the use of security questions is a method that can severely harm the security of the overall solution and the ways that this can be mitigated.

As shown by Rabkin [8], the widespread usage of security questions as a form of password reset mechanism can become a real threat, since the strength of such questions is based on the hardness of an information retrieval problem. As personal information becomes ubiquitously available online, the strength of security questions diminishes over time. One interesting result was the fact that there isn't (to time) enough evidence to support a widespread exploitation of such a preceived weakness on the usual targets, like the banking institutions. The most likely reason is the existence of multiple layers of auditing before actual money transfers take place, which makes this kind of attack less appealing than the more common ones, such as phishing. Clearly, when dealing with the recovery of blockchain private keys the threat is bigger, since the compromise of the private key can automatically mean the transfer of data, knowledge, or wealth.

Similarly, on the topic of the design of "good security questions", Jakobsson et al. [9] argue that there is a solution to finding suitable questions that should be based on long-lived personal preferences and knowledge, instead of publicly available information.

During the design of our approach, the results of both aforementioned studies, regarding the form and complexity of security questions, have been taken into account.

III. Methodology

The main goal of PKRS is to make a private key recoverable via a set of personal security questions. At first the user is asked a number of predefined questions and the answers are used to create a "secured private key". The original private key can be recovered once the user is able to answer correctly only a **subset** of the original questions.

This scheme tries to offer a trade-off between security, i.e. having an encrypted form of the original private key that can be stored safely in lower security environments, and usability, i.e. being able to recover the original private key after long periods of time, without the need of long and easily forgotten passphrases, or hardware security modules.

The core of PKRS is the SSS algorithm that allows to split a secret into chunks and recover the original secret, given only a subset of those chunks. In the next sections, we describe the creation of the secured private key and the recovery of the original private key, as can be seen in Figure III. We also discuss the design principles of the personal security questions, since they play a vital role in the hardening of the overall scheme.

A. Creation

Algorithm 1 shows the creation process of the secured private key. The required input of the algorithm is the following:

- private_key: The original private key that needs to be secured.
- questions: An array of strings that has been presented to the user as personal security questions.
- answers: An array of strings that contains the answers to the previous questions.

The minimum required number of questions is 3, even though the actual number of questions should be much higher, as we will discuss in a later section.

The algorithm initially takes the private key and splits it in a number of chunks (S[i]) equal to the number of questions. This is done via the SSS algorithm. Each S[i] is encrypted with a different key which is the answers (answers[i]) to the

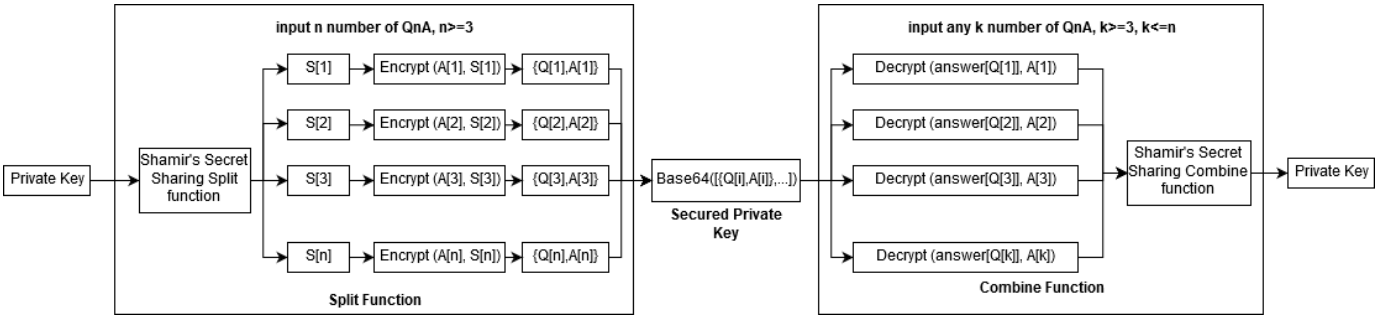

Fig. 1. Private Key Recovery Scheme

Algorithm 1: Create(private_key, questions, answers)

Result: sK: secured private key
Require: $questions.count \geq 3$
Require: $answers.count == questions.count$
1: $\mathbf{S} = SSplitFn(private\_key, questions.count)$
2: **for** $i = 0$ **to** $S.count$ **do**
3: $\quad \mathbf{A[i]} = Encrypt(answers[i], S[i])$
4: **end for**
5: **for** $i = 0$ **to** $A.count$ **do**
6: $\quad \mathbf{sK[i]} = \{q : Q[i], a : A[i]\}$
7: **end for**
8: $\mathbf{sK} = Base64Encode(sK)$

Algorithm 2: Recover(sK, answers)

Result: K: private key
Require: $answers.count \geq 3$
1: $\mathbf{sK} = Base64Decode(sK)$
2: $j = 0$
3: **for** $i = 0$ **to** $sK.count$ **do**
4: $\quad$ **if** $answers[i]! = null$ **then**
5: $\quad\quad \mathbf{S[i]} = Decrypt(answers[i], sK[i].a)$
6: $\quad\quad j++$
7: $\quad$ **end if**
8: **end for**
9: $\mathbf{S} = SCombineFn(S)$

aforementioned questions resulting in encrypted chunks A[i]. Finally, it creates an array of tuples that contain each question and encrypted answer (q:Q[i],a:A[i]) and encodes it in base64 resulting in the secured private key sK.

It should be noted that, in our implementation, we used AES as the encryption algorithm of step 3, although any symmetric algorithm can be used if needed.

B. Recovery

Algorithm 2 shows the recovery process of the original private key. The required input of the algorithm is the following:

- sK, The secured private key as created by Algorithm 1.
- answers: Any number of k answers to the questions q of sK.

The minimum required number of k is again 3.

The algorithm initially base64 decodes sK and produces the array of tuples containing the questions and encrypted chunks. Then, it iterates through those tuples and, if there is a given answer to the question of the current tuple, it decrypts the chunk using the answer as the key. Finally, it passes the decrypted chunks through the SSS combine function which produces the original private key.

C. Personal Security Questions Design

By now, it should be obvious that the personal security questions that will be used in PKRS are a risk factor that affect the whole scheme.

According to Rabkin [8], personal security questions can suffer from various weaknesses: 1) They can be "inapplicable" due to the fact that some questions presuppose a certain state of the user base that do not apply to the subjects (e.x., "What is the name of your spouse/husband"). 2) They can be "not memorable" and have answers that few individuals can reliably recall (e.x., "What is the middle name of you kindergarten teacher"). 3) They can be ambiguous in terms of too many correct answers or in terms of answers that can shift rapidly over time. 4) They can be "guessable" even without any knowledge about the subject, merely by social deduction (e.x. "How old did you marry" considering that a significant proportion of Europeans marry between 25-35). 4) They can be attackable via data mining of the subject's public information.

The last of the aforementioned weaknesses is the most commonly cited, due to the abundance of publicly available information one can mine from social networks and public repositories.

One interesting mitigating approach is the use of questions that "are chosen to relate to personal preferences rather than demonstrated actions and thereby avoid attacks based on data mining of public data to a large extent" [9]; to design the questions in such a way that they reflect long-term characteristics rather than short-term preferences. The goal is to "ask users for durable information they may not consciously remember and, are unlikely to have recorded in a public machine-readable form" [8].

Moreover, since questions of any short suffer from much

lower entropy than pass-phrases, another mitigating approach to overcome such inherent weakness is to substantially increase the number of available questions, and require a relatively small portion of correct answers.

Our approach can easily scale in such a manner, simply by configuring the number of chunks during the creation process and the number of required answers for a successful recovery. In the use case we will describe in section IV we use only a small amount of questions such as "Name 3 hobbies you would like to get into if you had the time and money" and "What do you do to unwind?" as a proof of concept. The full list of selected security questions can be found in the source code [1] .

Any real world implementation should contain a large pool of questions, and a threshold of at least 24 correct answers [9] required for successful recovery. All those questions should be designed based on the aforementioned principles.

IV. Use Cases - Self-sovereign Identity

In order to test and evaluate the validity of PKRS we have chosen to integrate it in a larger blockchain use case regarding digital identities.

The core characteristics of a blockchain are decentralization, security, immutability, and the absence of a central authority. Therefore it is widely considered as a technological foundation for implementing a SSI; a digital identity which is entirely controlled by its owner, without the need for a central service or provider. In such a use case, the private key becomes not only an access token to financial values, but a proof and representation of an (human) identity. Hence, suitable and usable recovery mechanisms become even more critical and relevant.

We have developed a working system for creating and managing SSIs and verifiable claims with the Ethereum blockchain acting as the primary infrastructure artifact. [2] : **SeSIS** (Self-sovereign Identity Service) The prototype supports three major features, related to identity management: self-registration, provision, and storage of identity attributes, and third-party verification of exactly these attributes. Our solution requires both, on- and off-chain technologies and incorporates different sub-components and services. The implemented use case is described in as follows: 1) A user issues an identity by using an SSI client, which abstracts the interaction with the Ethereum network and other services. This client could be locally installed, self-hosted, or offered by a provider. The user is not required to provide any personal information for the registration process. The registration process results in the local creation of a private and public key. 2) The SSI client interacts via a geth node with the Ethereum network and deploys an Identity Smart Contract (ISC) for the new user. The ISC itself represents the digital identity and exposes methods for adding attributes and corresponding verifications. In addition, a singleton contract, the Lookup Smart Contract (LSC), is used to link the address of the ISC to the address of the user. Hence, the LSC can be considered a global user registry. 3) The user enters personal attributes (e.g. date of birth or passport ID) through the SSI client. Since Ethereum is a public Blockchain, no sensitive or individual-related data is uploaded to the network in plaintext. Only the identifier of the attribute and a hashed and salted value of it are stored. For the actual data storage an encrypted Personal Data Store (PDS) (local or in the cloud) can be employed. 4) A verification authority (e.g. public administration) checks the claim of a user by applying the same hash function to the value of the attribute. A positive verification by the authority is indicated via the ISC.

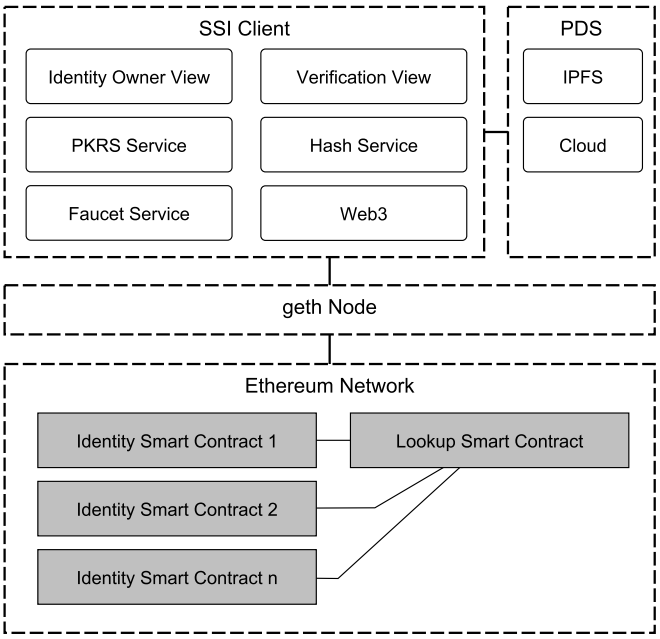

Fig. 2. High-Level Overview of the SSI Prototype

Our system implements a clear and meaningful use case for applying blockchain technology. However, we consider the technical handling of the private key a dealbreaker in successfully disseminating this use case. Average users do struggle with the management of the private key, whether it is a seed phrase, a hardware token, or a key file. To foster the awareness and practical adoption of a SSI, and avoid falling back to centrally managed workarounds, we applied PKRS as the primary authentication mechanism of our system. This profoundly lowers the entry barrier and increases the users confidence in dealing with the system.

As the user registers in step 1 of the workflow, they are presented with a wizard, asking the personal security questions. The answers are not stored or transferred, but only used as part of the PKRS process. After answering the questions, the secured private key (sK) is generated and presented to the user. The sK also needs to be stored, but this can be done in a less secure way, e.g., a cloud provider, hard drive etc.

[1] https://github.com/blockchain-werkstatt/shamir-library/blob/master/lib/config/config.js

[2] https://github.com/blockchain-werkstatt/identity-demo

In order to recover the access to the identity, the user has to provide the sK and correctly answer at least three of the personal questions.

We implemented the entire system and the described features to evaluate its feasibility. Fig. 2 shows an overview of the main modules and artifacts. The SSI client is written in JavaScript and employs the Vue.js framework[3] as the principal framework. As a PDS the InterPlanetary File System (IPFS)[4] can be used as dummy stand-in. A dedicated service is responsible to provide the users with required funds for interacting with the network (Faucet Service). For development and evaluation we are applying Ganache, a personal and local version of the Ethereum blockchain.[5].

The use case implementation demonstrates how the PKRS approach can be successfully integrated into a real-world blockchain application and how the registration and recovery processes is performed.

V. Conclusions

In this paper we have presented and evaluated our Partial Knowledge Recovery Scheme (PKRS) for securing and recovering a private key based on personal security questions. In PKRS, the users have to provide answers to a set of predefined personal security questions, which are used to encrypt the private key. A subset of the answers to these questions, and a generated secured private key, is required to recover the original private key. The design of the personal security questions follows best practices, to be sufficiently secure, answerable only by the user, and easy to remember. Our underlying algorithm is based on Shamir's Secret Sharing (SSS) and uses AES for symmetric encryption operations. We have successfully implemented the approach and integrated it into a blockchain use case for creating and managing self-sovereign identities. This enables the users to a much more usable login and recovery process than established schemes, like securing the private key or a mnemonic directly.

We positioned our solution and use case within Zooko's triangle [10] strongly on the sides *human meaningful* and *decentralized*. In addition, we aimed for the highest possible *security* within our scope of applying personal security questions as the means for authentication. As a result, our system offers a more natural and approachable process to the users.

The dissemination and adoption of blockchain applications is impeded by usability issues, essentially regarding the management of the underlying PKI infrastructure. Our solution, with an increased usability, has the potential to attract more users, lower the barriers, and create momentum for a broader application of blockchain and its potentials. A compromise in security is justifiable in the light of this broader objective. In addition, the security measures should fit the purpose. Our solution is suitable for use cases where transaction values are low to medium.

Our work is available as Open Source, and the Partial Knowledge Recovery Scheme (PKRS) can be integrated via an NPM package. [6]

VI. Outlook

Our solution could be applied to a variety of use cases, especially in the context of blockchain. Basically, every blockchain application requires authentication, which is based on private keys. Our next step is to identify an established and suitable blockchain application and integrate our approach for in-depth analysis. This includes an extensive user study to measure the actual acceptance, and, improve the personal security questions.

In addition, our approach can be applied to secure data sharing beyond blockchain. We will identify additional use cases where the security level of PKRS is appropriate.

References

[1] A. Shamir, "How to Share a Secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[2] Symantec, "How to: Split and rejoin pgp desktop 8.x keys," https://support.symantec.com/us/en/article.howto41916.html, Feb. 2012.

[3] M. Skibinsky, Y. Dodis, T. Spies, and W. Ahmad, "Decentralized storage of crypto assets via hierarchical shamir's secret sharing," https://github.com/vault12/whitepapers, Jan. 2018.

[4] D. Zage, H. Xu, T. M. Kroeger, B. Hahn, N. Donoghue, and T. R. Benson, "Secure distributed membership tests via secret sharing: How to hide your hostile hosts: Harnessing shamir secret sharing," *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–6, 2015.

[5] M. Ahmadvand., A. Scemama., M. Ochoa., and A. Pretschner., "Enhancing operation security using secret sharing," in *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications - Volume 4: SECRYPT, (ICETE 2016)*, INSTICC. SciTePress, 2016, pp. 446–451.

[6] S. Bartolucci, P. Bernat, and D. Joseph, "Sharvot: Secret share-based voting on the blockchain," in *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, May 2018, pp. 30–34.

[7] A. Miller, "Simple Security with Shamir Secret Sharing," https://blog.gridplus.io/simple-security-with-shamir-secret-sharing-15704166b8be, Aug. 2017.

[8] A. Rabkin, "Personal knowledge questions for fallback authentication: Security questions in the era of facebook," in *Proceedings of the 4th Symposium on Usable Privacy and Security*, ser. SOUPS '08. New York, NY, USA: ACM, 2008, pp. 13–23. [Online]. Available: http://doi.acm.org/10.1145/1408664.1408667

[9] M. Jakobsson, E. Stolterman, S. Wetzel, and L. Yang, "Love and authentication," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 197–200. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357087

[10] Z. Wilcox-O'Hearn, "Names: Distributed, Secure, Human-Readable: Choose Two," https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html, Nov. 2019.

[3] https://vuejs.org/

[4] https://ipfs.io/

[5] https://github.com/trufflesuite/ganache-cli

[6] https://www.npmjs.com/package/secretkeysharing