Privacy-preserving Revocation of Verifiable Credentials with Verifiable Random Functions

Athanasia Maria Papathanasiou
Department of Informatics
Athens University of Economics and Business
Athens, Greece
sissypapathanasiou@aueb.gr

George C. Polyzos
School of Data Science, CUHK-Shenzhen, China
ExcID P.C., Athens, Greece
Mobile Multimedia Laboratory, AUEB, Greece
polyzos@acm.org

***Abstract*—Self-Sovereign Identity (SSI) has highlighted the benefits and importance of granting users complete control over their identity. Unlike previous solutions, which entrust identity management to third-party applications or services, SSI empowers users to control their personal data. Nonetheless, compromised identities remain a challenge as they must be revoked in order to prevent additional privacy and security issues. However, in several SSI systems, privacy is compromised in favor of efficiency, allowing third parties to gain access to users' personal data. In this paper, we highlight the shortcomings of existing SSI solutions and propose a system that addresses credential revocation by balancing privacy with efficiency, by leveraging Verifiable Random Functions (VRFs).**

***Index Terms*—Self-Sovereign Identity, Proof-of-possession, Privacy, Personal data**

I. Introduction

Personal data control is of paramount importance since most existing digital identities are under the control of major technology companies or governments, resulting in potential privacy issues. *Self-Sovereign Identity* (SSI) systems introduce a user-centric approach to digital identity management. In an SSI system, users are able to select whom to share their personal data with and at which granularity. Moreover, SSI architectures introduce decentralization, as users control their personal data without having to rely on any trusted authority. Allen defined the principles of SSI, underscoring the significance of user control over their identities [1].

Despite the benefits that SSI systems introduce, there are still challenges that need to be addressed. One such challenge is related to compromised identities, which may be the result of theft or loss. In particular, in 2022, a total of 1802 instances of data compromises were recorded in the United States, while more than 422 million people experienced the consequences of data compromises, encompassing incidents such as data breaches, leaks, and exposure [2]. In some other cases, an individual's identity information may change or become outdated. If the revocation status is not updated, the individual's identity information may continue to be considered valid, even though it is no longer accurate. Due to the aforementioned reasons, revocation of credentials is required in order to avoid various security and privacy issues. However, revocation continues to pose a technical difficulty in SSI systems, as most proposed architectures sacrifice privacy for efficiency. Due to this limitation, it is important to design a revocation mechanism that will enable efficient revocation, while also ensuring that user privacy is not compromised. In this paper, we discuss the issues of revocation mechanisms in existing SSI systems and propose and evaluate a scheme that utilizes *Verifiable Random Functions* (VRFs), in order to handle revocation of credentials in a privacy-preserving manner, while also achieving a level of efficiency similar to a simple revocation list.

With the design of our solution, we make the following contributions:

- We separate the credential from the public key or *Decentralized Identifier* (DID) and instead include the output hash of the VRF, which makes it harder for third parties to connect the user with the credential.
- We use VRFs instead of digital signatures and therefore guarantee that the result is unique and verifiable. Note that with digital signatures, there may be multiple valid signatures for a given message. This is important as we need the verification algorithm to accept only one output for a given message.
- The VRF output is unique to a specific input and appears as random data. Also, the VRF proof doesn't disclose any user information, meaning that even if a malicious user steals the user's credentials, she cannot compute the same VRF output, because she lacks the secret key.
- We enable revocation of credentials by utilizing a hash table, which requires constant time in order to fetch the status of a credential, similarly to revocation lists, and thus efficient revocation is achieved.
- Finally, we enable *Proof-of-possession* (PoP), thus enhancing security and privacy.

The remainder of this paper is structured as follows: In Section II, we provide background information about the notions used in our study. In Section III, we discuss previous research in the same field, while in Section IV we outline our solution's design and implementation. We proceed to compare our solution with existing ones and discuss its benefits in Section V. Finally, we conclude our work in Section VI.

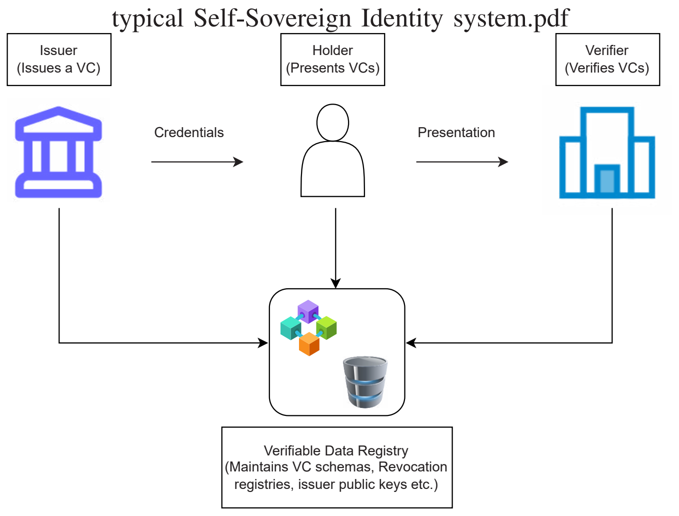
typical Self-Sovereign Identity system.pdf
Issuer (Issues a VC)
Holder (Presents VCs)
Verifier (Verifies VCs)
Credentials
Presentation
Verifiable Data Registry (Maintains VC schemas, Revocation registries, issuer public keys etc.)

Fig. 1. Verifiable Credentials Ecosystem

II. Background

A. *Verifiable Credentials*

SSI systems enable the creation of *Verifiable Credentials* (VCs). These are digital documents that utilize asymmetric cryptography in order to confirm the validity of a collection of claims about a Subject. VCs can represent the same information found in physical credentials but in a more secure and tamper-resistant way compared to their physical counterparts [3].

As illustrated in Figure 1, a standard SSI system comprises three main components: the Issuer, the Holder, and the Verifier. Initially, the Issuer generates a VC and sends it to the Holder. Upon receiving the VC, the Holder can produce a *Verifiable Presentation* (VP), which may consist of one or more VCs signed by the Issuer. Finally, the Verifier receives the VP from the Holder and cryptographically verifies its authenticity. It's worth noting that while in most cases, the Holder of the VC is also the Subject for whom the claims are made, sometimes the Holder can be different from the Subject (e.g., a parent holding a child's VCs). These operations are facilitated by a *Verifiable Data Registry* (VDR), such as centralized databases and *Distributed Ledger Technologies* (DLTs).

VCs rely on a set of established standards and technologies called the W3C Verifiable Credentials Data Model and Encoding [3], which sets the guidelines for organizing and encoding data to qualify as a VC. These standards facilitate the secure and interoperable exchange of information among various organizations and individuals. Moreover, this structure aims to ensure the privacy, security, and reliability of the data contained in the credentials.

B. *Verifiable Random Functions*

A VRF [4] is a cryptographic function that takes a secret key and an input x and produces an output that appears random and unpredictable, but can be efficiently verified by anyone with access to the corresponding public key. In other words, a VRF allows a party to generate a random-looking output that can be publicly verified as authentic, without revealing the underlying secret key. A more formal explanation of VRFs is the following:

- Key generation algorithm, which takes no input and outputs a public-private key pair (pk, sk).
- Compute $vrf\_prove(sk, x) \rightarrow p$, where sk is the secret key of a user and x is a known input. A hash can also be extracted from the proof using $vrf\_hash\_from\_proof(p) \rightarrow h$. The value h is the VRF output, which appears random and unpredictable to anyone without knowledge of the secret key sk. The proof p is a cryptographic proof that allows anyone to verify the authenticity of the output h, given the public key pk.
- A third party can verify the correctness of the output using $vrf\_verify(pk, x, p) \rightarrow h'$, where pk is the public key of the user with secret key sk. If h' is equal to h then the proof p is valid.

The security properties of a VRF are the following:

- Collision resistance: It is computationally infeasible to find two inputs that produce the same VRF output.
- Pseudorandomness: The VRF output h should appear random and unpredictable to anyone without knowledge of the secret key sk.
- Verifiability: Given the public key pk, the input x and the proof p, it should be computationally efficient to verify the authenticity of the output h, without knowledge of the secret key sk.

III. Related work

We divide revocation mechanisms of VCs in the following categories:

- Centralized revocation: In this approach, a central entity manages the process of revoking credentials by maintaining a list of credentials that have been invalidated. Verifiers examine this list before accepting a credential to ascertain whether it has been revoked. One example of a system employing this method is Let's Revoke [5], and a comparable approach is outlined in a W3C draft [6]. However, these solutions may introduce privacy concerns due to the use of unique credential identifiers, which allow Verifiers to potentially link the Holder to the specific credential. In this paper, we extend the W3C VC data model in order to incorporate the proofs and hashes of VRFs and also describe the additional computations that need to be performed in order to allow inclusion of VRF operations.
- Decentralized revocation: Examples of decentralized revocation mechanisms utilize DLTs [7] or smart contracts [8], [9], typically by storing a revocation list inside a blockchain. However, it is important to note that these approaches can potentially introduce privacy concerns due to their reliance on blockchain technology for handling revocation [10]. Furthermore, this category includes *Peer-to-peer* (P2P) networks. Chotkan *et al.* [11] propose

a protocol based on gossiping to disseminate revocation information through a P2P network, which could lead to increased communication overhead. Stokking *et al.* [12] introduce an SSI system that uses direct P2P communication, again introducing communication overhead.

- Zero-knowledge proof-based revocation: In this method, a Verifier can check the validity of a credential disclosing only essential information and thus achieving a high level of privacy. Typical examples include dynamic accumulators [13], which enable the verification of a group of values without exposing any details about each specific value within the group. IRMA[1] and Sovrin [14] leverage dynamic accumulators in order to achieve revocation of credentials. However, these systems significantly lack efficiency compared to revocation lists.

IV. System design and Implementation

In Figure 2, an overview of our system's architecture is depicted. In our solution, each VC has an identifier which is the same for all credentials of the same type. For instance, all VCs that represent a driver's license have the same identifier, which is different from the identifier of VCs that represent a Bachelor's degree. We also use the functions vrf_prove, $vrf\_hash\_from\_proof$ and vrf_verify, as discussed in Section II. More specifically, we utilize VRFs for two purposes. First, the Holder can prove that he owns a specific VC. Second, we achieve PoP as the Holder can prove that he possesses the private key associated with the public key. Note that the public key or DID itself is not included in the VC. PoP is achieved using only the hash of the VRF, which is included in the credential. Moreover, revocation is enabled using a hash table with the hash output of the VRF as keys and bit values 0 or 1, with 0 representing non-revoked credentials and 1 referring to revoked credentials.

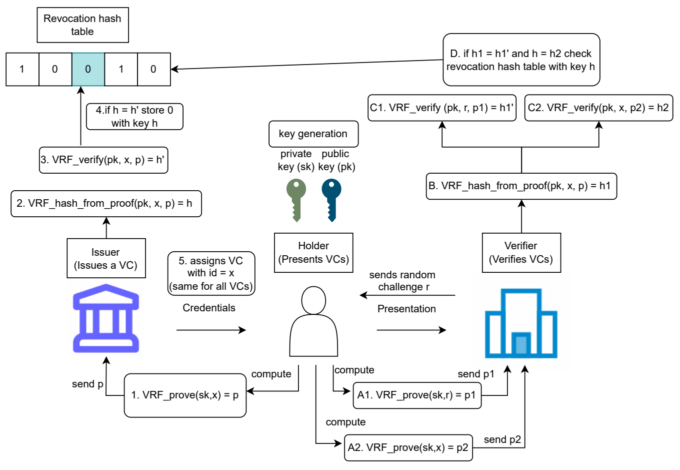
Revocation hash table
1
0
0
1
0
D. if h1 = h1' and h = h2 check revocation hash table with key h
4.if h = h' store 0 with key h
C1. VRF_verify (pk, r, p1) = h1'
C2. VRF_verify(pk, x, p2) = h2
key generation
private key (sk)
public key (pk)
3. VRF_verify(pk, x, p) = h'
B. VRF_hash_from_proof(pk, x, p) = h1
2. VRF_hash_from_proof(pk, x, p) = h
Issuer (Issues a VC)
5. assigns VC with id = x (same for all VCs)
Holder (Presents VCs)
Verifier (Verifies VCs)
sends random challenge r
Credentials
Presentation
send p
1. VRF_prove(sk,x) = p
compute
compute
send p1
A1. VRF_prove(sk,r) = p1
compute
A2. VRF_prove(sk,x) = p2
send p2

Fig. 2. System architecture

From a high level perspective, our system works as follows. Initially, the Issuer signs and sends a VC to the Holder. Prior to this step, a couple of checks need to be performed. First, the Holder has to compute $vrf\_prove(sk, x)$, where sk is her secret key and x is the VC identifier, which is the same for all credentials of the same type (e.g., license to drive), in order to prove that she possesses the private key associated with the public key she submitted to the system. The output of $vrf\_prove(sk, x)$ is a proof p, which is sent to the Issuer. When the Issuer receives p, he computes $vrf\_hash\_from\_proof(p)$, which outputs a hash h and then calculates $vrf\_verify(pk, x, p)$, which outputs h' with pk being the public key of the Holder. He then checks whether there is a registration with key h in the revocation hash table and if not, he places bit 0 with h as key. Finally, if h' is equal to h, the Issuer includes the hash h inside the VC(steps 1-5 in Figure 2).

When the Holder wants to present one or more VCs to the Verifier, a valid VP needs to be constructed. The Verifier also sends a random challenge r to the Holder, so as the latter can use it for PoP. Subsequently, the Holder calculates $vrf\_prove(sk, r)$, which outputs p_1. He also calculates $vrf\_prove(sk, x)$, where x is the identifier of the VC we mentioned above. This outputs p_2, which will be used to prove that the Holder indeed possesses the specific VC. Note that the Holder needs to compute $vrf\_prove(sk, x)$ for every VC he wants to include in the VP. After these calculations, p_1, p_2 and the public key of the Holder are included in the proof section of the VP (steps A1 and A2 in Figure 2).

Once the Verifier receives the VP he extracts the proof section and computes $vrf\_hash\_from\_proof(p_1)$, which outputs h_1, $vrf\_verify(sk, r, p_1)$, which outputs h'_1 and $vrf\_verify(sk, x, p_2)$, which outputs h_2. If h_1 is equal to h'_1 and h_2 is equal to h, which is the hash included in the VC we mentioned above, then the Holder indeed possesses the specific VC and the secret key associated with the public key. The Verifier then checks the revocation table using h_2 as key and if the VC status has a value equal to 0 then the VP validation has been performed successfully (steps B, C1, C2 and D in Figure 2).

When a VC needs to be revoked, the Issuer can simply place bit 1 in the revocation hash table. In case an adversary steals the VC of a Holder, he cannot compute a valid proof for the hash inside the VC, as he does not possess the secret key. We could also enable decentralized revocation by storing the hash table inside a blockchain. Although this would offer transparency, it needs to be checked in terms of efficiency.

The code[2] for our solution is implemented in Python programming language using Flask. It includes examples of VCs, which can be used to construct VPs and makes all the necessary checks we mentioned in this section. For the implementation of the VRF functions we used libsodium library[3] of the Algorand blockchain[4], where the VRF operations are based on an elliptic curve-based VRF construction designed

[1]https://irma.app/docs/what-is-irma/

[2]Code available at: https://github.com/sissyp/VRFsForVcRevocation

[3]Code available at: https://github.com/joshjdevl/libsodium-jni-algorand

[4]https://algorand.com/

Revocation mechanism	Complexity	Privacy
Revocation lists [5]–[7]	$O(1)$	Allow Verifiers to correlate the Holder with the VC
P2P networks [11], [12]	$O(n)$	Revocation based on trusted relationships between nodes
Dynamic accumulators [13], [14]	$O(n)$	Do not expose any detail about each VC
VRFs	$O(1)$	Decouple VC from public key or DID

TABLE I
COMPARISON OF REVOCATION MECHANISMS

by Goldberg *et al.* specified in an IETF draft[5]. These functions are described similarly to the background information we provided in Section II. Finally, we created a wrapper Python library called VRFLibrary, which can be used to call the VRF functions using the C written functions of the original library.

V. EVALUATION

In Table I we compare our system with existing ones in terms of complexity and privacy. Regarding complexity, in our system we need constant time to fetch a VC from the revocation hash table, which is also the case in revocation lists. In P2P network architectures revoking a credential needs $O(n)$ time in the worst case scenario, where n is the size of the network. However, it is expected that on average, revocation updates will not propagate through the whole network and thus the complexity would be logarithmic. Moreover, dynamic accumulators require $O(n)$ time, meaning that each time we need to revoke or generate a VC, we have to compute again the witness value of the accumulator. Our system, as well as systems that utilize lists for VC revocation can be implemented with a central registry in the Issuer, or a decentralized one in a blockchain. However, including a hash table in a blockchain will not be as efficient as a list with VCs with unique identifiers.

In terms of privacy, our solution decouples the VC from the Holder's public key or DID and only includes the hash of the VRF, therefore making it harder for Verifiers to correlate the Holder with the VC and track its digital presence, as can happen in solutions that include the public key or DID. Even if a malicious user steals the VC of a Holder, he will not be able to compute the same hash as he does not possess the secret key. Furthermore, we replace the use of digital signatures in the Holder's side with VRFs, whose output is unique to a specific input and indistinguishable from random data. Digital signatures are not unique for the message they sign and thus when privacy is considered a major requirement, VRFs are more suitable. Also, we enable PoP, in which the random challenge that the Verifier sends to the Holder could be the same for a certain number of users and thus resulting in a more private solution. P2P networks, which are based on trust relationships to achieve revocation are less preferred compared to a system that is based on cryptographic primitives. Lastly, although dynamic accumulators are considered the best solution in terms of privacy, each time a credential is revoked or generated, the witness value needs to be calculated again and transmitted to the other parties.

[5]https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-vrf-03

VI. CONCLUSIONS

We leveraged VRFs in order to build a revocation mechanism for VCs, which are structured according to the W3C Data Model. Our solution achieves efficient revocation by utilizing a revocation hash table. We also replaced digital signatures during the VP, leading in unique provability for a specific input, while at the same time enabling PoP, which results in higher privacy levels. We compared our system in terms of complexity and privacy with existing ones and showed that our solution is efficient and feasible considering existing standards, while also offering a certain level of privacy.

REFERENCES

[1] Allen C. The Path to Self-Sovereign Identity; 2016. Accessed 2023-09-15. Available from: http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html.

[2] Johnson J. Annual number of data breaches and exposed records in the United States from 2005 to 2020; 2021. Available from: https://www.statista.com/statistics/273550/.

[3] Sporny M, Longley D, Chadwick D. Verifiable credentials data model 1.0. W3C Candidate Recommendation. 2019 Mar.

[4] Micali S, Rabin M, Vadhan S. Verifiable random functions. In: 40th annual symposium on foundations of computer science. IEEE; 1999. p. 120-30.

[5] Smith T, Dickinson L, Seamons K. Let's Revoke: Scalable global certificate revocation. In: Network and Distributed Systems Security (NDSS) Symposium; 2020. .

[6] Sporny M, Longley D. Revocation List 2020: a privacy-preserving mechanism for revoking Verifiable Credentials; 2021. Accessed: 2023-09-15. Available from: https://w3c-ccg.github.io/vc-status-rl-2020/.

[7] Xu J, Xue K, Tian H, et al. An identity management and authentication scheme based on redactable blockchain for mobile networks. IEEE Transactions on Vehicular Technology. 2020;69(6):6688-98.

[8] Tariq A, Haq H, Ali S. Cerberus: A blockchain-based accreditation and degree verification system. IEEE Transactions on Computational Social Systems. 2022.

[9] Lundkvist C, Heck R, Torstensson J, Mitton Z, Sena M. uPort: a platform for Self-Sovereign Identity; 2016. Accessed: 2023-09-15. Available from: https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221.pdf.

[10] Yli-Huumo J, Ko D, et al. Where is current research on blockchain technology?—a systematic review. PloS One. 2016;11(10):e0163477.

[11] Chotkan R, Decouchant J, Pouwelse J. Distributed Attestation Revocation in Self-Sovereign Identity. In: 47th Conference on Local Computer Networks (LCN). IEEE; 2022. p. 414-21.

[12] Stokkink Q, Ishmaev G, Epema D, Pouwelse J. A truly self-sovereign identity system. In: 46th Conference on Local Computer Networks (LCN). IEEE; 2021. p. 1-8.

[13] Camenisch J, Lysyanskaya A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Annual International Cryptology Conference. Springer; 2002. p. 61-76.

[14] Khovratovich D, Law J. Sovrin: digital identities in the blockchain era;. Accessed: 2023-09-15. Available from: https://sovrin.org/library/sovrin-digital-identities-in-the-blockchain-era/.