Blockchain Redaction in Self-Sovereign Identity

Šeila Bećirović*, Špela Čučko†, Muhamed Turkanović†, Haris Šupić*, and Saša Mrdović*
*Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina
†Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia

***Abstract*—The development of blockchain has allowed for the development of new concepts and ideas. A completely immutable ledger might not be appropriate for all new applications that are being envisaged for the blockchain. One of them is self-sovereign identity. The aim of this paper is to analyze the possible use cases for blockchain redaction in SSI. Main concepts of redaction and a summary of the current research progress are given. Use cases for redaction in SSI are categorized and described alongside their existing solutions. Detailed proposal for possible use cases is given and comparison is drawn between this solution and existing solution. Future challenges are introduced.**

***Index Terms*—blockchain, blockchain redaction, self-sovereign identity**

I. Introduction

Complete control over one's digital identity is an approach that garners more traction every day. Today, the concept can be found in an identity management model known as self-sovereign identity (SSI), that is relatively new, and as such it still has certain issues that remain unresolved, e.g. revocation, lost device, right to be forgotten and other. SSI is built using distributed ledger technology, commonly blockchain. With the rapid growth of blockchain usage, the need for changing or erasing the data on it became evident, and so blockchain redaction was introduced. [1].

Using review and analysis as primary methodologies, in this paper we introduce a theoretical approach to the possible usage of blockchain redaction for the SSI ecosystem. Prior to this paper, usage of redaction in SSI has not been comprehensively discussed. The aim of this work is to give the first introduction of blockchain redaction as a possible solution to certain issues of SSI. We identified and defined potential use cases for blockchain redaction in SSI. Besides the use cases, we define opportunities and challenges that can arise from using blockchain redaction as a solution to SSI issues.

In section II SSI, blockchain and blockchain redaction will be explained. In section III improvements over the original redaction will be presented. In section IV potential use cases and existing solutions in SSI will be presented. In section V a discussion concerning the benefits and possible issues will be given. In the final section, a conclusion will be drawn and a proposal for future research will be given.

II. Preliminaries

A. Blockchain

Blockchain is a public or private distributed ledger built on a peer-to-peer network [2]. It enables agreements on transactional data, data sharing across a network of untrusted participants, without relying on a central trusted authority. Blockchain clients that run on blockchain network nodes verify and store transactions on the ledger. The ledger itself represents a continuous append-only database of transactions. In order to achieve trust, a majority of nodes needs to reach a consensus on transactional data states. The data structure of a blockchain is a list of identifiable blocks that store a predefined maximum number of transactions and are cryptographically linked to the previous block. The blocks form a chain [3]. Distributed ledger technologies (DLT) such as Blockchain can be used as a part of an ecosystem for SSI.

One of the main characteristics of blockchain is immutability. It is thought to be impossible to change or erase data from the blockchain, especially if it is public and permissionless. In certain scenarios this can be seen as a limiting characteristic for the usage of blockchain, and so due to the necessity for removal of sensitive and inappropriate content, blockchain redaction was introduced.

B. Self-Sovereign Identity

Self-Sovereign Identity (SSI) is an identity management model where each digital identity is controlled, and managed by the entity to which the identity and related data belongs [4]. Each user can freely control and manipulate their digital identity, including personal information, receive and collect verifiable credentials from the issuers and choose which information they want to share without the reliance on any external authority. Identity management infrastructure is somewhat decentralized, and this eliminates a single point of failure and enhances security, trust, and privacy. Self-Sovereign identity is considered to be in its infancy, and there are few standards and architectures that define it completely [5]. The development of SSI was initiated with the developments of decentralized technology, specifically blockchain technology, although the later is not always needed for the SSI implementation [6].

SSI consists of elements such as decentralized identifiers and verifiable credentials. Before further explaining certain issues in SSI, SSI elements should be explained. W3C defines decentralized identifiers (DIDs) as a "new type of identifier that enables verifiable, decentralized digital identity. A DID refers to any subject (e.g., a person, organization, thing, data model, abstract entity, etc.) as determined by the controller of the DID." DIDs are URIs that associate a DID subject with a DID document allowing trustable interactions associated with that subject. DID document represents a set of data describing the DID subject, including mechanisms, such as cryptographic public keys, that the DID subject or a DID delegate can use to authenticate itself and prove its association with the DID.

Verifiable credential is a standard data model and representation format for cryptographically-verifiable digital credentials as defined by the W3C Verifiable Credentials specification [7].

While implementing SSI elements, such as decentralized identifiers (DIDs) and verifiable credentials (VCs), SSI sometimes relies on the distributed ledger such as blockchain to enable everyone in the network to have the same source of truth i.e., DLTs are thus used as the trust anchor, e.g., stored hashes of DIDs or exchanged VCs. Furthermore, in some government frameworks like the European SSI Framework (ESSIF), DLTs are used for various registries, e.g., trusted issuer registry, DID registry etc [8]. Using the blockchain, participating SSI entities (identity holders, issuers and verifiers) can additionally verify someone's identity and credentials. With SSI development in early stages, there are still several open questions, e.g. how to administer a revocation list, and how to retrieve access if a device with the wallet is lost or stolen. The ability to remove a revoked element from the distributed registry (blockchain) can become a useful feature and the solution for mentioned issues in SSI implementations. One of the possible solutions for such requirements is blockchain redaction, which represents a novel technique, with its own advantages and challenges.

C. Blockchain Redaction

Ateniese, et al. in the first papers about redaction [1] defines redaction as one of the following "(i) re-writing one or more blocks, (ii) compressing any number of blocks into a smaller number of blocks, (iii) and inserting one or more blocks not at the end of the ledger." Redactions can be made only by authorized entities and under specific constraints; moreover, redactions are publicly auditable by existing miners since they must approve the redacted block. [1].

Removal of improper and illegal content, creting a re-writable storage for smart contract and overlay application, enabling the "right to be forgotten" and consolidation in financial institutions are just some of the examples where redactable blockchain is needed [1].

In the blockchain, a block is defined in its triple form $B = (s, x, ctr)$, where $s \in \{0,1\}^k$ denotes the hash of previous block, $x \in \{0,1\}^*$ denotes the block content and $ctr \in \mathbb{N}$ is the proof of work of the block. Block B is valid if:

$$validblock_q^D(B) := (H(ctr, G(s,x)) < D) \wedge (ctr \leq q) = 1. \tag{1}$$

Here, $H : \{0,1\}^* \rightarrow \{0,1\}^k$ and $G : \{0,1\}^* \rightarrow \{0,1\}^k$ are collision-resistant hash functions, and the parameter $D \in \mathbb{N}$ is the block's difficulty level. The parameter $q \in \mathbb{N}$ is a bound that in the Bitcoin implementation determines the size of the register ctr; here it is arbitrary and represents the maximum number of hash queries that a user is allowed to make in any given round of the protocol. Blockchain is simply a chain or sequence of blocks C. The rightmost block is known as the head of the chain $Head(C) := (s, x, ctr)$. Any chain can be extended to a longer chain by attaching a valid block $B' := (s', x', ctr')$, such that $s' = H(ctr, G(s,x))$. The new head of the chain becomes the latest block [9].

TABLE I
REDACTION IN PERMISSIONED AND PERMISSIONLESS SETTINGS [1]

Permissioned setting	**Permissionless setting**
The algorithm takes a chain and a set of indices that represent the position of the blocks that are going to be redacted as the input. The algorithm receives a chameleon hash trapdoor key.	The algorithm takes a chain and a set of indices that represent the position of the blocks that are going to be redacted as the input. The trapdoor key is secretly shared amongst a fixed set of users, that are in charge of redacting the blockchain.
Computation of collision is performed by central authority, and the content of the block is replaced, which creates a new chain with the replaced block. This chain will replace the original.	A set of users engage in a secure multiparty computation protocol (MPC) to complete the algorithm in a fully distributed manner. Using a secret sharing scheme, users receive a share of the trapdoor key and reconstruct it. The secret sharing scheme needs to cope with the possibility of corrupt users submitting incorrect shares. Each user will then calculate the collision and construct the chain with the replaced block.
Central authority broadcasts this chain as a special chain, meaning that every user should adopt it.	The redacted chain will be broadcast as a special chain that should replace any other chain.

The main idea for redaction (presented in Figure 1) is to set the inner hash function (function $G(s,x)$) in blockchain to be a chameleon hash function. Chameleon hashing was introduced by Krawczyk and Rabin [10]. A chameleon hash is a cryptographic function that contains a trapdoor. Using the trapdoor, collisions are easily generated, but without it, collisions are hard to find. Re-writing the content of each block is possible by finding collisions in the hash function. Now, a new block is defined as $B := (s, x, ctr, (h, \xi))$, where the new component (h, ξ) is the hash/check pair for a chameleon hash. It should be noted that the new content of the redacted block is irrelevant, and as such represents either deletion or a change of the data. The algorithm is different for permissionless and permissioned settings [1].

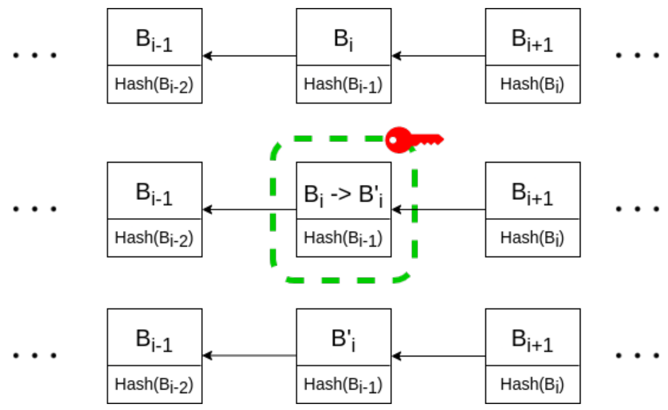

Fig. 1. Redaction in blockchain [11]: (1) Blockchain without redaction; (2) Redaction of middle block; (3) Redacted blockchain - links remain the same.

The basic algorithm for redaction in permissione and permissionless settings are given in table I.

The original concept of redaction is focused on the permissioned setting, while in permissionless, it suffers from scalability issues because the trapdoor key is shared [12].

III. Related Works

Improvements over the original redaction method have been made in papers [12]–[15].

Puddu et al. [13] presented a redactable blockchain called μ chain. The sender of the transaction encrypts different versions of the transaction, and the decryption keys are secretly shared among miners. One transaction among them is decrypted, and is considered active. If the user wants to change the transaction, he sends a request, and according to the policy, the appropriate transaction is decrypted and becomes a new active transaction. Redaction of a transaction allows for a more detailed redaction over the existing redaction of an entire block. This is beneficial to the redactions concerning one user's transaction data.

In [14], Deuber et al. introduced the redactable blockchain protocol in the permissionless setting. When a redaction is proposed by a user, a consensus-based voting period starts. After obtaining enough votes, a redaction is performed. This redaction is limited to certain transactions, such as *OP_RETURN*. One of the core concepts of SSI is that it should not rely on any external authority. In that case, a permissionless redaction is required. Redaction still needs to be a strictly defined and controlled process that does not allow the redaction of relevant used registries and similar.

In [15], Ashritha et al. propose improvements over the original redaction by using the Chameleon hash function that enables modification of a block without changing other blocks' contents. They propose splitting the trapdoor key among major validators and reconstructing it using Multi-Party Computation. They also propose the creation of a second trapdoor key that will be in the sole possession of the creator of the block. This trapdoor key will be used when the block redaction should not happen without the consent of the creator. User's transactions should be redactable with their permission.

Jing Xu et al. [12] propose an instantaneous redactable blockchain protocol. They present a generic approach to designing a redactable blockchain protocol in the permissionless setting. It is applicable to both proof-of-stake (PoS) blockchain and proof-of-work (PoW) blockchain. According to the existing experiments, redaction and verification in the blockchain are instantaneous. Instantaneous redaction brings benefits to redaction in SSI, especially in the cases of revocation and GDPR compliance.

The only mention of redaction in SSI is in [16]. Authors propose an identity management and authentication scheme based on redactable blockchain for mobile networks. Blockchain is used to record the public keys of legitimate users, while redaction is used to delete users' information. Through experimental results, the scheme can reduce the revocation and communication overhead.

Changing previous information in SSI is introduced in [17], [18]. These solutions do not use blockchain as a DLT. In paper [17], authors introduce reclaimID, a SSI solution built on the decentralised GNU Name System (GNS) in combination with ciphertext-policy ABE using type-1 pairings. They define the operation of deletion as a removal of the attribute and removal of authorized access by using attribute tag version. Revocation is done through attribute versioning as well. When access of a requesting party to an attribute is revoked, they increment the attribute version and publish the encrypted attribute value to the name system. In [18], authors propose a model in which social media platforms such as Facebook and LinkedIn are used as a means of requesting, generating, and revoking credentials, along with claim presentation and revocation of presented claims. Only attribute disclosure is supported in this model.

Even though there exist different platforms to build SSI on, blockchain remains the most popular one, so finding solutions for SSI issues using blockchain is required.

IV. Overview of use cases for redaction in SSI

When we are considering the use cases for redaction, we must observe what is stored on the ledger. From it, we determine that the use cases can be divided into two categories. The first category is from the issuer's perspective. Issuer redaction includes possible redaction of data related to public DIDs, VC (e.g., revocations), public keys revocation, and legal guardianship/delegation/controllership redaction. The second category is from the holder's perspective, i.e., for user-specific issues, such as device revocation, holder's data, DIDs, GDPR compliance, right to be forgotten, etc. It should be noted that holder and issuer roles can apply to both the individual user and an institution. An overview of the categories is given in Table II and an overview of the process of redaction is given in Figure 2.

TABLE II
An overview of categories of redaction in SSI

Holder redaction	**Issuer redaction**
'Right to be forgotten'	Registry/VC revocation
Agent revocation	Public keys revocation
User data revocation	Revocation of delegation

Redaction in SSI should be "instantaneous" and can be done differently in permissioned and permissionless settings. In permissioned settings, redaction can be done by sending a request to the authorized user or group of users. They will proceed with the required block redaction, as previously explained in section II.

In permissionless settings, the initial proposer of the transaction in a block could have the ephemeral trapdoor key that corresponds to a particular block. To redact, the initial proposer or entity related to the transaction will send a request to all validators, which were randomly selected to prevent malicious attacks. Validators will perform a secure multi-party computation to regenerate the secret trapdoor key. Using the trapdoor key, redaction will be completed. As a result of the redaction, a new chain with the redacted block will be created and accepted as a valid one. One block will be added for tamper-evident logging.

In the following sections, SSI issues and their current solutions are explained. Usage of redaction is introduced as a possible solution to the issues.

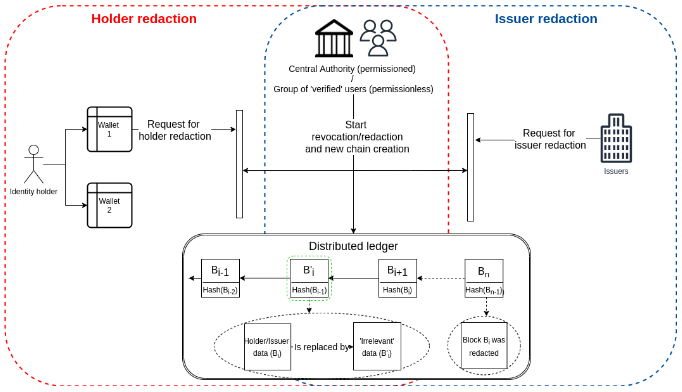
Holder redaction
Issuer redaction
Central Authority (permissioned)
/
Group of 'verified' users (permissionless)
Wallet 1
Wallet 2
Identity holder
Request for holder redaction
Start revocation/redaction and new chain creation
Request for issuer redaction
Issuers
Distributed ledger
B_{i-1}
$Hash(B_{i-2})$
B'_i
$Hash(B_{i-1})$
B_{i+1}
$Hash(B_i)$
B_n
$Hash(B_{n-1})$
Holder/Issuer data (B_i)
Is replaced by
'Irrelevant' data (B'_i)
Block B_i was redacted

Fig. 2. An overview of the redaction process

A. Issuer redaction

1) Public key/DID redaction: DIDs represent identifiers and they do not provide information about the subject. They are used in combination with VCs where they prove to the third parties that DID subject has ownership of certain attestations or attributes and/or that the issuer of VC is also a DID subject. The third-party can then use the presented cryptographically protected proof to verify the ownership and trustworthiness of the claims about the subject.

Public DIDs and public keys are sometimes stored on the blockchain. They belong to entities that should be publicly identifiable and as such they do not have a complete self-sovereignity (government institutions, ministries, certain companies, etc). When an entity loses its legal status and the legal right to issue VCs, its publicly available keys and DIDs should be revoked by using redaction. By redacting the publicly available information on the blockchain, or rather, replacing it with irrelevant data, attestations from the issuer will no longer be possible, and every VC issued by them will not be valid.

2) Registry/VC redaction: Revocation of a credential includes deleting or updating it. When information inside a credential is no longer valid or is changed, the credential should be revoked, and a new credential should be created. Current revocation solutions include: (i) time-revocation: credentials contain an expiration time, (ii) revocation list: credentials are linked to an index of a revocation registry, which can only be updated by the credential issuer, or (iii) a proof of non-revocation: credentials include the zero-knowledge capability to prove that the credential has not been revoked [8], [19], [20]. Time-revocation is usable when a VC has an expiration date. In the case of revoking a VC without an expiration date, time-revocation is not usable. Revocation lists are an obvious solution to the revocation problem. However, they are not privacy preserving, since credentials have to be presented in a way that they can be correlated to the revocation list, and by doing so they can also be correlated to their presenter [19].

Revocation of credentials is currently done by using revocation registries and cryptographic accumulators in Hyperledger Indy [19]. The revocation registry is a mathematical concept. We can represent it as a list of numbers (factors) where each number has an index in a row (tails file). Each of these numbers is assigned to one VC in a way that every VC has its unique number. All the numbers 'multiplied' together are the so-called accumulator (e.g., merkle trees). Verifying a VC requires a calculation of the accumulator value on the blockchain, using the accumulator value without the VC row and the value of the corresponding VC row. When a VC is revoked, the number is removed from the registry, and the value of the accumulator changes. The issues of this type of revocation are speed, and size of the tails file. This type of revocation adds complexity to issuance, proving, and verification. Currently it is not a feature of W3C's Verifiable Credentials, since W3C's VC standard enforces no revocation method, but rather the need for a revocation mechanism and its requirements [19], [21].

We propose a following solution to this problem. When a VC is issued, it is hashed with the public key of the issuer and stored alongside its details on the blockchain. As a form of verification, besides the signature proof and the issuer verification through the public blockchain-based registry, the computation of the VC's hash can be used. If the calculation matches the one on the blockchain, the VC is verified, otherwise, it is not. This concept is simple and more straightforward than having a constantly updated accumulator value and a tails file. When a VC is revoked, instead of having a large tails file with a corresponding removed row and a new accumulator value, only the hash of the VC on the blockchain is redacted (replaced with irrelevant data). In that way, when a VC hash is calculated, it won't match the one on the blockchain; and the VC is revoked. In the case of revocation, or rather redaction, that hash and the details will no longer be on the ledger. Visual representation is given in 3.

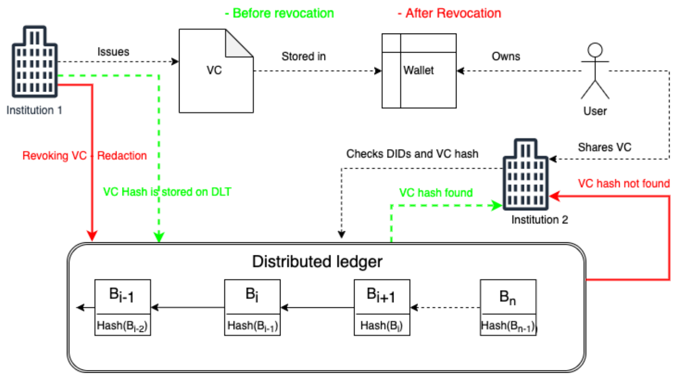

Fig. 3. Example of VC revocation

3) Guardianship/Delegation/Controllership redaction: A case can be made for redaction in revocation of delegation, guardianship, and controllership. Sovrin defines the roles as following: (i) a delegate acts on behalf of the delegator, (ii) guardian represents dependent, (iii) thing controller controls things [22]. For delegation Sovrin proposes the use of their technology stack, as they argue that the layers of SSI infrastructure called the Trust over IP (ToIP) [23] are uniquely suited to support digital guardianship. They combine underlying layers of cryptographic or "technical" trust with higher layers of human trust as represented by legal, business, and social frameworks. The first layer consists of the public blockchains or other decentralized networks, which support DIDs. The second layer is defined by SSI digital wallets, agents, and hubs that speak the DIDComm protocol to establish peer-to-peer, DID-to-DID connections for secure communications and data exchange. A guardian/delegate/controller has VCs and DIDs in its wallet that confirm the legal status of the relationship between the parties. On layer three, human trust enters the ToIP stack in the form of the "trust triangle" among issuers, holders, and verifiers of digital credentials (VCs concerning delegation/guardianship/controllership). The fourth layer is the layer where human governance is added to the first three layers. The only technology at this layer is the definition of a small set of special verifiable credentials used by governance authorities to publish governance credentials and define legal grounds for relationships [22].

In the case of termination of the guardianship, delegation, or controllership, redaction can be used for revocation. Proof of transfer of control should be stored on the blockchain as a transaction that combines DIDs of the controller/guardian/delegate with the DID of the other party. Information such as DIDs and VCs that confirm the legal status is still in the wallet of the guardian/controller/delegate. For status verification, information from the blockchain is used. In the case of revocation, guardianship/delegation/controllership can be completely removed from the blockchain or can be changed on the blockchain, ensuring that verification will fail.

B. Holder redaction

1) 'Right to be forgotten' and user data redaction: Let's take a look at the use case for user data and the right to be forgotten redaction. In compliance with the GDPR, a person has a right to have their data completely removed. In the context of blockchain immutability, no personally identifiable information should be stored on the blockchain. As such, only public elements of the DIDs (public keys) are stored on the blockchain and eventual hashes in different registries. Peer DIDs not stored on the ledger are just removed from the device. If an institution with public DIDs decides to delete their identity, alongside every associated interaction stored on the ledger (public DIDs, hashes in registry), redaction could be used. The existing public DID information would be removed.

2) Device redaction: Let us consider the deletion of an agent. Users should be able to create their identity and use it on multiple devices, be it phones or laptops. In their wallet app, users store all their private data, which includes their VCs. In case of a stolen phone, someone might abuse the data on the phone and use it for illicit activities. For such cases, a way to make the information on the device useless is required. One such way is to use agent authorization policies or registering devices of one's identity.

In Hyperledger Indy/Sovrin, each identity has a list of used devices/agents, and they play a role in the authentication using public/private keys of the device. To recognize and revocate user's devices, Sovrin is developing agent authorization policies, which combine cryptographic accumulators and zero-knowledge proof cryptography. In case of a stolen or a lost device, we need a way to delete the device and prevent someone from using the stolen device or identity. When a device is stolen or lost, the user could use another authorized device to write on the blockchain that his mobile phone's authorization is now revoked. Revoking device authorization includes revoking the existing relationship keys of the device with a wallet [24].

Using redaction, we can envision the following use case. For each identity, new device is registered on the blockchain. Each device contains its pair of keys, of which the public key is stored on the blockchain. When a user sends a verifiable credential, the device's keys are used to encrypt it. This way, we make sure that the appropriate device is the one that is sharing information. In case of a lost/stolen device, a user can begin the redaction process. The user can sign in using a different device and send a request for the revocation of the stolen device. If the user has only one device and one wallet, there is a need for a specific device code that only the user knows. When he starts the process of revocation, that code is required to confirm which device is getting revoked. The device's public keys are redacted, and thus the data is rendered useless. When someone sends a verifiable claim using the device, it will be rejected due to the key pair not matching.

V. Discussion

Work of Xu, et al. in [12] propose the usage of redaction for user revocation, specifically fine-grained dynamic illegal user revocation method instead of revocation lists. The authors propose that if some users access network illegally, network operators can revoke these users using redaction i.e. removing illegal user's registration, and prevent them from accessing network. We expand on this solution by introducing more possible use cases for redaction, beside user removal. Our proposal differs from this one in the terms of having an illegal user that registers on the network (such user does not exist), but the concepts of using redaction instead of revocation lists remains.

As we go through the use cases, we see that there is merit to blockchain redaction. Its merit comes from simplifying the existing processes. Using redaction, we avoid breaking users' privacy, unlike some solutions such as revocation lists. Additionally it provides us a possible solution to the problem of user data appearing on the ledger. The processes for revoking public keys/DIDs are simplified. We know that they are stored on the blockchain, and redacting them represents their revocation. The same can be said with registry/VC redaction. If a verifiable hash of the VC is kept on the ledger, redacting it makes it non-verifiable. This removes the complexities from the issuance, proving and verification. There is no need for a large tails file. User privacy is preserved. A possible solution for controllership, guardianship and delegation is shown. It is based on storing certain information on the ledger, which are removed when the relationship ends. By using redaction, the problem that arises with GDPR compliance, and more specifically, the 'Right to be forgotten', in SSI is solved. Removing everything related to the user on the blockchain is possible with redaction. Using redaction, the transaction correlating to the VCs with their hashes are no longer available. Unfortunately, already shared, now revoked data can still suffer from an analog-loop, and be locally-persisted, but in the SSI system it is not verifiable. One possible solution for a lost device used for SSI identity is to use redaction for its public key. In that way, the device becomes unusable.

Since this process changes the otherwise immutable data source, it can be dangerous if not used correctly. As such, it can only be done by a form of centralized authority or a specifically chosen set of users. It should not be performed lightly and by everyone. That is why there should always be a trace, in a form of a new transaction, that there was a change on the blockchain. Solving every problem with redaction and enabling redaction of every transaction may completely disrupt the entire system (i.e. removal of registries and similar). As such, redaction should be used in strictly defined manner on a specific set of problems. In this paper we defined a subset of these problems that can benefit from the usage of redaction.

There are alternatives to blockchain distributed ledger for SSI implementation. They include Hashgraph, Iota Tangle and R3 Corda. Both Iota and Hashgraph use Directed Acyclic Graphs (DAGs) as an alternative data structure for maintaining the ledger. Each of the mentioned DLTs has a proposed SSI solution [25]–[27]. Details regarding revocation of credentials and removal of publicly available information are an active research topic. Some explored ideas include adding a new transaction as an update concerning revocation [25], or using the time-based revocation method. These DLTs maintain immutability, and no data can be removed or rewritten on the ledger.

In this paper, we have shown that not maintaining immutability has its merit. Two questions remain: (i) Is it possible to have redaction in other DLT technologies? (ii) How do we achieve interoperability between different SSI solutions if one uses redaction?

VI. Conclusion

Even though redaction is a fairly new concept still in development, a case can be made for its use in developing Self-Sovereign Identity (SSI). With each new data registry added to the SSI, a new case for redaction is created. The instantaneous redaction can reduce the existing overhead of revocation and even communication. To integrate redaction in SSI, thorough performance examination of different redaction types in an SSI environment is required. Beside the redaction in blockchain, redaction in other DLT technologies needs to be examined.

References

[1] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain–or–rewriting history in bitcoin and friends," in 2017 IEEE European symposium on security and privacy (EuroSP). IEEE,pp. 111–126, 2017

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system,", Decentralized Business Review, p. 21260, 2008.

[3] F. Tschorsch, and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 2084–2123, 2016.

[4] K.C. Toth, and A. Anderson-Priddy, "Self-sovereign digital identity: A paradigm shift for identity," IEEE Security & Privacy, vol. 17, no. 3, pp. 17–27, 2019.

[5] Š. Čučko, and M. Turkanović, "Decentralized and self-sovereign identity: Systematic mapping study," IEEE Access, pp. 1–1, 2021.

[6] D. Van Bokkem, R. Hageman,G. Koning, L. Nguyen, and N. Zarin, "Self-sovereign identity solutions: The necessity of blockchain technology," arXiv preprint arXiv:1904.12816, 2019.

[7] [Online]. Available: https://www.w3.org/TR/did-core/

[8] [Online]. Available:https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=379913698

[9] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in Annual international conference on the theory and applications of cryptographic techniques. Springer, pp. 281–310, 2015

[10] H. Krawczyk, and T. Rabin, "Chameleon hashing and signatures," 1998.

[11] K. Rajasekhar, S.H. Yalavarthy, S. Mullapudi, and M. Gowtham, "Redactable blockchain and it's implementation in bitcoin," International Journal of Engineering Technology, vol. 7, no. 1.1, pp. 401–405, 2018.

[12] J. Xu, X. Li, L. Yin, Y. Lu, Q. Tang, and Z. Zhang, "Redactable blockchain protocol with instant redaction." IACR Cryptol. ePrint Arch.,vol. 2021, p. 223, 2021.

[13] I. Puddu, A. Dmitrienko, and S. Capkun, "µ chain: How to forget without hard forks," IACR Cryptology ePrint Archive 2017/106, 2017.

[14] D. Deuber, B. Magri, and S.A.K. Thyagarajan, "Redactable blockchain in the permissionless setting," in 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019, pp. 124–138.

[15] K. Ashritha, M. Sindhu, and K. Lakshmy, "Redactable blockchain using enhanced chameleon hash function," in 2019 5th International Conference on Advanced Computing Communication Systems (ICACCS). IEEE, 2019, pp. 323–328.

[16] J. Xu, K. Xue, H. Tian, J. Hong, D.S. Wei, and P. Hong, "An identity management and authentication scheme based on redactable blockchain for mobile networks," IEEE Transactions on Vehicular Technology, vol. 69, no. 6, pp. 6688–6698, 2020.

[17] M. Schanzenbach, G. Bramm, and J. Schutte, "reclaimid: Secure, self-sovereign identities using name systems and attribute-based encryption," in 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).IEEE, 2018, pp. 946–957.

[18] G. Lax, and A. Russo, "A lightweight scheme exploiting social networks for data minimization according to the gdpr," IEEE Transactions on Computational Social Systems, vol. 8, no. 2, pp. 388–397, 2021.

[19] "0011: Credential revocation." [Online]. Available: https://hyperledger-indy.readthedocs.io/projects/hipe/en/latest/text/0011-cred-revocation/README.html

[20] [Online]. Available: https://w3c-ccg.github.io/vc-status-rl-2020/

[21] "How identity revocation on the blockchain works," 10 2020. [Online]. Available: https://tykn.tech/identity-revocation-blockchain/#WhatisRevocation

[22] "Guardianship whitepaper," 12 2019. [Online]. Available: https://sovrin.org/library/guardianship-white-paper/

[23] M. Davie, D. Gisolfi, D. Hardman, J. Jordan, D. O'Donnell, and D. Reed, "The trust over ip stack," IEEE Communications Standards Magazine, vol. 3, no. 4, pp. 46–51, 2019.

[24] "What if i lose my phone?" 05 2019. [Online]. Available: https://sovrin.org/library/lost-phone/

[25] J.F. Millenaar, and M. Yarger, "The case for a unified identity," https://files.iota.org/comms/IOTATheCaseforaUnifiedIdentity.pdf

[26] L. Baird, M. Harmon, and P. Madsen, "Hedera: A public hashgraph network & governing council," White Paper, vol. 1, 2019.

[27] "The case for self-sovereign identity," 04 2020. [Online]. Available: https://www.r3.com/blog/the-case-for-self-sovereign-identity/