Sovrin: digital identities in the blockchain era

Dmitry Khovratovich
University of Luxembourg

Jason Law
Evernym, Inc.

***Abstract*—In this paper we describe a practical digital identity project of a global scale, which solves a number of privacy and scalability problems using the concepts of anonymous credentials and permissioned blockchains. Even though both ideas have been known and developed for a number of years, our system appears to be the first amalgamation. The prototype has been developed, tested, and published as open source.**

I. Digital identity: history and challenges

The idea of digital identity can be traced to the beginning of the computer era, when the user profiles were first transferred across networks. In the real world, different authorities issue licenses, passports, identification cards, credit cards, checks, etc., generally called *credentials*, to ordinary users for their presentation to various service or goods providers. This brings the concept of Issuer, User, and Verifier as subjects of the identity system.

It immediately became clear that issuance, storage, and presentation procedures must follow a number of security requirements to be competitive with the traditional paper credentials:

- *Compatibility.* Credentials issued by different Issuers can be combined for a single presentation by User who is required to prove several properties simultaneously: e.g. that he is at least age 18 employed by XYZ Corp. with a driver license and good eyesight (does not require corrective lenses).
- *Unforgeability.* A malicious User is unable to present a credential not issued by a legitimate Issuer.
- *Scalability.* A system must handle hundreds of Issuers and billions of Users, such as an international passport system. It might be required to handle thousands of transactions per second like the Visa system [1].
- *Performance/Low latency.* Verification of credentials must be almost instant and not require broadband network connectivity per credential, even if it is issued by a previously unknown Issuer. Credentials can be stored and processed easily by the User even if he is offline.
- *Revocation.* An Issuer can revoke any credential such that a User and any Verifiers can know within a reasonable amount of time that credential was revoked.

Additionally, a digital identity system could provide features impossible or difficult to implement in a paper identity system:

- *Minimal dependencies.* Aside from an earlier issuance, an Issuer should not be involved during the preparation and presentation of proofs, and the verification of credentials, including proof of non-revocation.
- *Privacy/Anonymity.* The real identity of a User can be kept secret from Verifiers who do not need it.
- *Unlinkability.* Different presentations of the same credential cannot be linked together.
- *Selective disclosure.* Any subset of attributes embedded in a credential can be kept secret from Verifiers.

The latter group of properties is particularly attractive for Users from authoritarian countries or those who are concerned about leaks of their data from careless Issuers or Verifiers.

The privacy-oriented digital identity schemes have been proposed as U-Prove [2] (mainly Microsoft, primarily based on Brands' credentials [3]) and Idemix [4] (mainly IBM, primarily based on the Camenisch-Lysyanskaya credentials [5]), with the ABC4Trust and FIDO [6] projects encompassing these two. Despite significant effort expended on both projects, neither is in widespread use today.

We deduce that all these systems lack compatibility and scalability, as an actual implementation would require a meta-system that congregates multiple Issuers and Verifiers and manages credential schemas. Before recently, all such system would require a (trusted) global third party for the exchange and distribution of Issuers' data and parameters. It seems that such a centralized party on a global scale could not exist.

II. Our contributions

We designed and implemented a system called **Sovrin**, which integrates anonymous credentials with revocation [7], [8] for privacy, unforgeability, performance, unlinkability and a distributed ledger, taking best practices from Ethereum [9] and BFT protocols.

The implementation is mostly finished and will go live in the near future.

A. Anonymous credentials for privacy

It is well known that the anonymity and unlinkability properties are provided by various kinds of *anonymous credentials*, the concept dating back to seminal works by David Chaum in 1980s [10] and refined in a number of follow-up papers including well-known [3], [7], [11]. Zero-knowledge proofs allow a User to prove the possession of a credential without showing the credential itself, thus providing unlinkability. Additional features include delegation [11] and revocation [8].

We reference the Idemix specification [4], [12] as the basis for our anonymous credential module as it provides unlinkability by default (in contrast to U-Prove). Although Idemix is available as a Java library, for portability and fast prototyping we select the Charm framework [13], which provides a Python API for large integers, signature schemes, and pairings. It also allows us to integrate the pairing-based revocation module easily.

B. Our treatment of the revocation feature

The revocation feature was missing in the original papers behind U-Prove and Idemix and was added later in different ways. The revocation procedure assumes that each credential has special Revocation-ID attribute with value i_R, and Issuer can at any moment revoke a particular i_R. For Verifiers this means that if a User with a revoked i_R presents his credential, this can be detected by the Verifier.

We observe that in addition to various drawbacks described above, the revocation feature provides quite a *privacy leak*. Indeed, consider a typical attack setting: a Verifier colludes with an Issuer to disclose the identity of a User. The Issuer then revokes half of the issued credentials, and the Verifier asks the User to update his revocation data. If the User succeeds in presenting a valid non-revocation proof, he is among the users with non-revoked i_R. Then the Issuer reinstates the earlier revocations and revokes another group of revocation IDs of equal size. Repeating this procedure $\log N$ times, the Issuer-Verifier team can uniquely identify one of N users.

C. Our selection of revocation methods

Since 2001, the following revocation methods have been proposed:

- *Signature lists* [14]. Issuer regularly releases a list of signatures, one per non-revoked i_R. User proves that his credential has the same ID as in some signature. Verification is fast, but the amount of data retrieved by User and computation he does is huge, and multiple presentations are linkable.
- *Cryptographic accumulators* [7], [8]. The issued i_R is added to (or removed from) a constant-size accumulator A, and a User must prove that his (non-disclosed) ID is still contained in A. This approach is reasonably fast for all parties.
- *Forward revocation lists* [15]. For each revoked i_R, an Issuer generates a revocation entry $f(i_R)$. A User presents a non-revocation proof to a Verifier, who then tests the proof against every entry in the list. If all matches fail, the credential is deemed valid. This method requires significant computational resources from the Verifier.

We decided to use accumulators based on bilinear maps [8] for performance reasons. The small order of the target group allows for short accumulators and witnesses. We have our own implementation of accumulators, which is based on a corrected version of paper [8]. The drawback is that the User must be aware of all revoked credentials, as the proof is updated every time another credential is revoked and needs (public) Issuer-specific data for the update. If User has small capacity, the data he needs to update the non-revocation proof reveals his ID. Existing research papers suggest User connecting to Issuer every time before creating a non-revocation proof [8], which is privacy-vulnerable if Issuer or his agent colludes with Verifier.

D. Revocation with attribute-based sharding.

The accumulator-based revocation procedure requires the User to maintain his non-revocation data w using the indices $\{i_R\}$ of the credentials revoked and issued from the last update. To update w the User also needs to access a global set $G = \{g_i\}$ of *validity tails*, published by Issuer. The tail set G is immutable but should be generated in advance and occupy quite big space (256 bytes per credential). Even though the number of tails needed for the update is proportional to the number of revoked credentials in the period, the tails retrieved by the User effectively disclose his own index i_R and thus his identity. For a global Issuer the User would be unable to carry the entire set of tails at his device, as they would require GBytes of storage.

To prevent the privacy leak, we propose to partition the credential IDs $\mathcal{I}$ into *shards* $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_s$ of limited size so that the tail set $G(\mathcal{I}_j)$ for each shard would be feasible to download and carry (say, a dozen of MBytes). Together with the credential to present, the User will then inform the Verifier of his shard number j so that the Verifier uses the corresponding accumulator data A_j.

To make the shard index privacy-oblivious, we suggest *attribute-based sharding*, when Issuer selects q attributes $\{a_i\}$ of necessary granularity (year of birth, zip code, etc.) and partitions the credential set by each a_i independently. For each attribute a_i and each shard an accumulator $A_{i,j}$ will be published. Nevertheless, since accumulators are of constant size, the total amount of space consumed by $q \times s$ accumulators is negligible. At the time of presentation the User selects the attribute on which he will prove the non-revocation. This attribute should be among those that are already disclosed to the Verifier. Before presentation, User downloads the entire set $G_{i,j}$ of tails that correspond to his share j at attribute i. As the tails are signed, he ask the Verifier to download, and this does not reduce the privacy.

We also implement the revocation-liveness parameter. Verifier specifies in the presentation definition how old accumulators he accepts for non-revocation proofs. To prevent the revocation attack as above we recommend a user to reject any definition that requires an accumulator younger than a day old. In this case for a million of users the Issuer-Verifer collusion pair would need at least $20 = \log 2^{20}$ days to identify a user.

III. Blockchain ledgers for scalability

A. Background

Blockchain is a concept of distributed database with limited trust between the peers. It has recently appeared as a merge of two mostly unrelated concepts: cryptocurrencies without trusted third parties and synchronization in a distributed database.

The first ingredient, the distributed consensus, was motivated by network databases of a single enterprise, which receive updates from different clients and needs consensus on the order. Faulty database nodes can be handled by fault-tolerant (FT) protocols [16], [17] and malicious nodes by Byzantine fault tolerant (BFT) protocols [18]. The BFT protocols tolerate up to $1/3$ malicious nodes, which is a reasonable bound if the nodes are few and under certain (though not full) control of the system architects. With dozens of nodes, the existing BFT protocols demonstrate throughput of thousands transactions per second [19]. Among the BFT protocols one emphasizes the most well known PBFT [20] and its refinements Aardvark [21], Zyzzyva [22], RBFT [23], Stellar [24], etc. Although many of these protocols come with a security proof, it is often quite involved and difficult to verify. The stress tests are another way to compare the protocols, and it was shown that PBFT has a very slow recovery process.

The second ingredient, the cryptocurrencies, date back to electronic coins proposed in works by Chaum in 1980s [25], [26]. He showed how to setup a system with arbitrarily many atomic coins so that parties could exchange them and prove ownership while respecting some privacy. However, it required a trusted third party for certain steps and a protocol that would enable the parties to communicate and store their credentials or logs if needed. Bitcoin [27] introduced storage of all coin transactions in a public and distributed ledger (blockchain) and issuance (mining) of coins being a result of transactions processing. To make the emission smooth and competitive, the Bitcoin protocol stores the transaction blocks not by mutual agreement but by the proof of certain computation done (PoW - proof of work), which must be proportional to the total computing power of the network. This effectively prevents any group of peers to attack the protocol unless they have a computational fraction of 25% or higher [28]. However, the maximum throughput is limited by a few transactions per second for so big a network to synchronise.

B. *Our take on blockchain-based ledgers*

The primary objects we store and update on the ledger are

- Public User pseudonyms;
- Issuer credential definitions and public keys;
- Revocation updates.

As we do not store actual presentations of identity, the transaction rate in the system is not expected to be high, and a few (up to a hundred) validating nodes suffice. As long as the nodes are added primarily to provide fast access and geographic variety, the node holders do not need motivation to run the nodes in the exchange of coins. Then the transaction processing does not have to be competitive, and BFT protocols with higher throughput are a good choice. We note that there also exist hybrid BFT-PoW protocols [29], but we have not found them useful for distributed identity management.

We use an Ethereum-based [9] ledger, which records transactions and root hashes of the Merkle tree over the state of public pseudonyms, Issuer public keys, revocation data, credential definitions, etc.. The immutable data such as revocation tails (see below) are stored off-chain in distributed file systems such as IPFS with relevant links from the ledger state.

For the consensus protocol we have chosen the BFT family of protocols as we limit the number of nodes to a few hundred, impose restricted membership (thus permissioned blockchain) and have partial control over many of them.

Our BFT protocol is called *Plenum* [30], and it is an enhancement of RBFT [23], which was chosen its resilience and fast recovery properties and implemented it. We replaced MACs with EdDSA signatures [31] as very fast implementations now exist, designed a leader election protocol, and added new blacklisting strategies.

We designate a group of nodes called *Validators* to agree on transactions and to sign the ledger blocks so that Users that are not currently connected to the ledger can request their Agents to retrieve the ledger data and verify it against certain block hash.

We do not store the anonymous credentials and secret keys; it is the User job to keep them in a secure storage in his smartphone or PC. As traditional in Idemix-like protocols, User generates a master secret and embeds it in all the credentials he gets signed from Issuers. Later on he can prove that all the credentials are chained with this secret.

The overview of the protocol is depicted in Figure 1.

References

[1] “Visa inc. at a glance,” 2015, https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf.

[2] “U-prove,” 2012, https://www.microsoft.com/en-us/research/project/u-prove/.

[3] “Rethinking public key infrastructures and digital certificates, phd thesis,” 1999, available at http://www.credentica.com/the_mit_pressbook.html.

[4] “Specification of the identity mixer cryptographic library version 2.3.0,” 2009, http://domino.research.ibm.com/library/cyberdig.nsf/papers/EEB54FF3B91C1D648525759B004FBBB1/$File/rz3730_revised.pdf.

[5] J. Camenisch and A. Lysyanskaya, “A signature scheme with efficient protocols,” in *SCN*, ser. Lecture Notes in Computer Science, vol. 2576. Springer, 2002, pp. 268–289.

[6] “Fido alliance: Uaf protocol,” 2014, https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.pdf.

[7] J. Camenisch and A. Lysyanskaya, “Dynamic accumulators and application to efficient revocation of anonymous credentials,” in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 2442. Springer, 2002, pp. 61–76.

[8] J. Camenisch, M. Kohlweiss, and C. Soriente, “An accumulator based on bilinear maps and efficient revocation for anonymous credentials,” in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, vol. 5443. Springer, 2009, pp. 481–500.

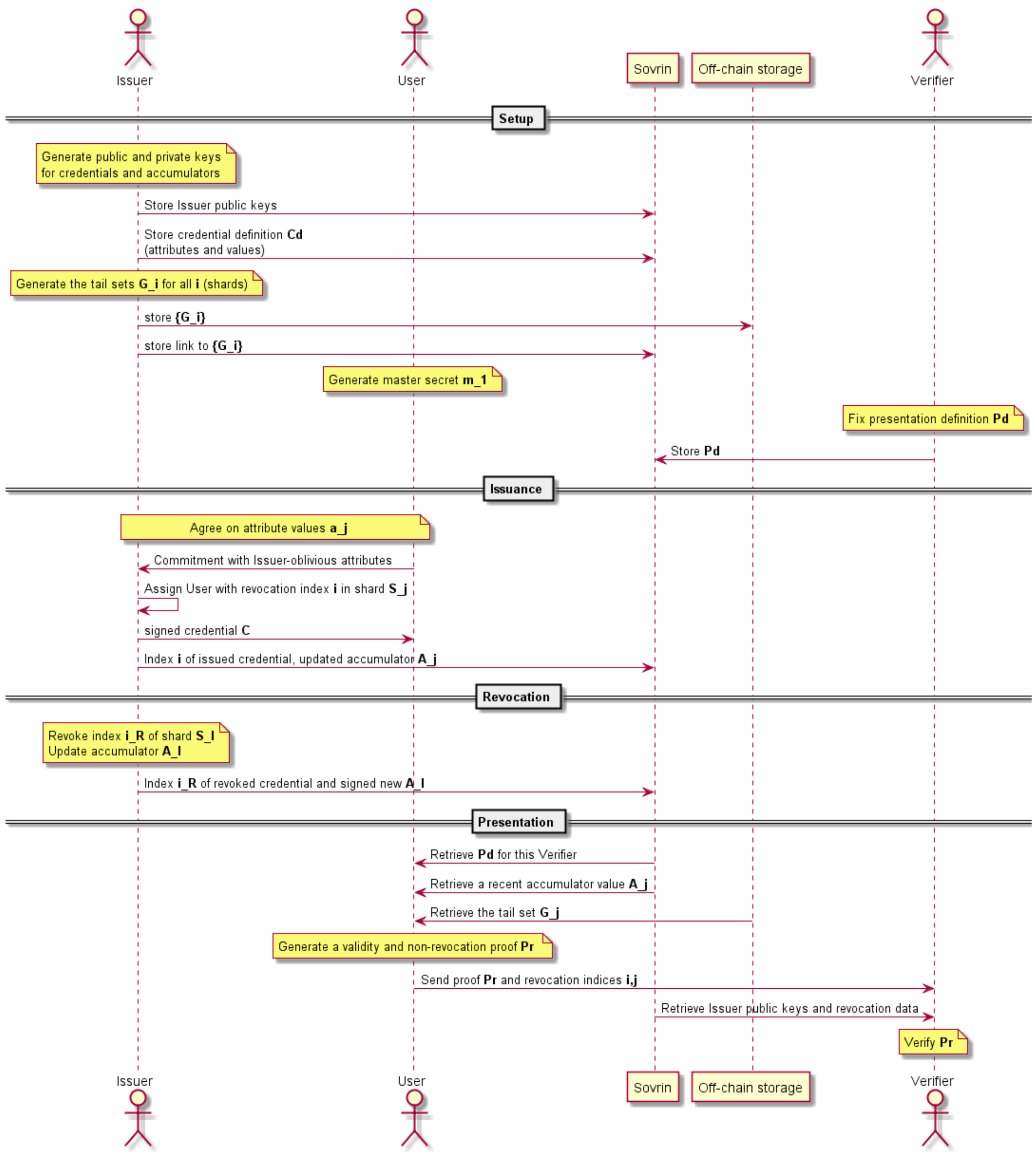
Issuer
User
Sovrin
Off-chain storage
Verifier
Setup
Generate public and private keys for credentials and accumulators
Store Issuer public keys
Store credential definition Cd (attributes and values)
Generate the tail sets G_i for all i (shards)
store {G_i}
store link to {G_i}
Generate master secret m_1
Fix presentation definition Pd
Store Pd
Issuance
Agree on attribute values a_j
Commitment with Issuer-oblivious attributes
Assign User with revocation index i in shard S_j
signed credential C
Index i of issued credential, updated accumulator A_j
Revocation
Revoke index i_R of shard S_l
Update accumulator A_l
Index i_R of revoked credential and signed new A_l
Presentation
Retrieve Pd for this Verifier
Retrieve a recent accumulator value A_j
Retrieve the tail set G_j
Generate a validity and non-revocation proof Pr
Send proof Pr and revocation indices i,j
Retrieve Issuer public keys and revocation data
Verify Pr
Issuer
User
Sovrin
Off-chain storage
Verifier

Fig. 1. Overview of credential presentation integrated with Sovrin.

[9] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014, http://gavwood.com/paper.pdf.

[10] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[11] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Randomizable proofs and delegatable anonymous credentials," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 5677. Springer, 2009, pp. 108–125.

[12] J. Camenisch and E. V. Herreweghen, "Design and implementation of the *idemix* anonymous credential system," in *ACM Conference on Computer and Communications Security*. ACM, 2002, pp. 21–30.

[13] "Charm: A tool for rapid cryptographic prototyping," 2016, http://charm-crypto.com/.

[14] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki, "Revocable group signature schemes with constant costs for signing and verifying," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, vol. 5443. Springer, 2009, pp. 463–480.

[15] E. R. Verheul, "Practical backward unlinkable revocation in fido, german e-id, idemix and u-prove," *IACR Cryptology ePrint Archive*, vol. 2016, p. 217, 2016.

[16] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998.

[17] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *USENIX Annual Technical Conference*. USENIX Association, 2014, pp. 305–319.

[18] L. Lamport, R. E. Shostak, and M. C. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[19] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains. master thesis," 2016, https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf?sequence=7.

[20] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI*. USENIX Association, 1999, pp. 173–186.

[21] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making byzantine fault tolerant systems tolerate byzantine faults." in *NSDI*, vol. 9, 2009, pp. 153–168.

[22] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. L. Wong, "Zyzzyva: Speculative byzantine fault tolerance," *ACM Trans. Comput. Syst.*, vol. 27, no. 4, 2009.

[23] P. Aublin, S. B. Mokhtar, and V. Quéma, "RBFT: redundant byzantine fault tolerance," in *ICDCS*. IEEE Computer Society, 2013, pp. 297–306.

[24] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Draft, Stellar Development Foundation, 15th May, available at: https://www. stellar. org/papers/stellarconsensus-protocol. pdf (accessed 23rd May, 2015)*, 2015.

[25] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 403. Springer, 1988, pp. 319–327.

[26] D. Chaum, "Online cash checks," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 434. Springer, 1989, pp. 288–293.

[27] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009, http://www.bitcoin.org/bitcoin.pdf.

[28] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *EuroS&P*. IEEE, 2016, pp. 305–320.

[29] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," *IACR Cryptology ePrint Archive*, vol. 2016, p. 199, 2016.

[30] "Plenum: Byzantine fault tolerant protocol," 2015, https://github.com/evernym/plenum.

[31] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, "High-speed high-security signatures," *J. Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, 2012.