Industry-Grade Self-Sovereign Identity

On the

Rowdy Chotkan, BSc.
MSc Student
TU Delft
R.M.Chotkan@student.tudelft.nl

Introduction

Since the dawn of the Information age, digital trust has been an issue requiring many workarounds. The core concepts of the internet are simply not built with trust in mind; there exists no standardised identity layer (Cameron, 2005). As a result, the current landscape of identification and authentication mechanisms form a digital ecosystem of "digital one-offs" (Cameron, 2005). As a consequence, the popularity of these digital one-offs by early pioneers of the Internet has resulted in an oligopoly in digital identity of Big Tech companies. This oligopoly results in an asymmetrical control of digital identities held by Big Tech. Wherein a regular oligopoly, consumers are at a disadvantage price-wise (Stigler, 1964), in this technical oligopoly, these identity providers have an asymmetrical control of ones digital presence and the ability to nullify access to such services in case one violates their terms of service. In addition, this oligopoly results in large information asymmetries as Big Tech has increasing amounts of knowledge on their users.
Furthermore, increasing needs for digital identities from governments such as the European Union, has catapulted the research and relevancy of the field itself. With the State of the Union Address by President Von der Leyen stating that:

"*Every time an App or website asks us to create a new digital identity or to easily log on via a big platform, we have no idea what happens to our data in reality. That is why the Commission will soon propose a secure European e-identity. One that we trust and that any citizen can use anywhere in Europe to do anything from paying your taxes to renting a bicycle. A technology where we can control ourselves what data and how data is used.*"

This need of digital identity furthermore stems from urgency of COVID-19 vaccination passports, requiring digital verifiability and validity across borders. As a result, this digital and socio-economical gap can prove to be filled by the concept of *Self-Sovereign Identity* (SSI).

SSI aims to fill the gap in digital trust by providing verifiable digital identities, putting the user at the center. SSI is an issue requiring multiple state-of-the-art technologies to be realised, thus, the feasibility of developing a schema that is both technologically and usability-wise sound, can be proven to be hard. Numerous solutions exist (e.g. Sovrin[1] and Serto[2], however, many require proprietary technologies or hardware, or require specialised infrastructure limiting equality in the network. As SSI combines multiple technologies, such as decentralised ledger infrastructure, public key infrastructure, and secure data management, many of the existing solutions do not stem strictly from academia, making their results more difficult to reproduce and limiting the analysis of their design choices.

This article introduces *Industry-Grade Self-Sovereign Identity*: a purely academic Self-Sovereign Identity framework focusing on an open standard, with intrinsic equality across the network and an offline-first design. The scheme is based on the previous works by Stokkink and Pouwelse (2018), Stokkink, Epema, and Pouwelse (2020) and builds upon the IPv8 protocol stack (Halkes & Pouwelse, 2011; Zeilemaker, Schoon, & Pouwelse, 2013). The main contributions of this work are a functioning SSI scheme, which can be said to be of *industry-grade*. IG-SSI makes the following contributions to the work set out by Stokkink and Pouwelse (2018): (1) Trusted Authority (TA) concepts, (2) offline verification capabilities, (3) revocation mechanisms (referred to as the Hybrid Revocation Modal), (4) improved security considerations (5) improved usability considerations (6) a reference implementation of the semantic layer, and (7) a reference implementation showcasing practical use-cases.

Design

Attestations

Attestations can be said to be the core concept of Self-Sovereign Identity. With attestations, we refer to cryptographically signed data, enabling verification of information through validation of signatures. In other words, an client, i.e. an Authority, cryptographically signs—attests—information for another party, the Subject. Allowing any third-party, a Verifier, to verify that the data was attested to by the Authority. As becomes apparent from this description,

[1]For Sovrin, see: `https://sovrin.org/`

[2]For Serto, see: `https://www.serto.id/`

these roles are neither mutually exclusive nor static: a single party can both be e.g. the Authority and the Subject for an attestation, whilst being solely a Verifier in another instance.

Asymmetric Encryption

A rather straightforward realisation of Attestations can be achieved through asymmetric encryption and signatures. For instance, using public key encryption, an Authority can, through the use of his private key (SK), encrypt the hash of a plaintext message (m) and the public key of the Subject (PK), resulting in $e(\mathcal{H}(m|pk))$. This allows any party that knows the corresponding public key of the Authority, to verify that the data m was attested to by the Authority for the Subject. There, however, exist several limitations with this approach. Firstly, disclosing the attestation and, thus, verifying the signature always reveals the corresponding plaintext values. This is not desirable, as the attestation may comprise sensitive data. Secondly, this would disclose more information than is necessary. For instance, verifying where on is of age of majority, should not require to disclose one's actual age. Rather, proving that one is above said threshold should suffice in such an instance. As such, Zero-Knowledge Proofs (ZKP) may prove to overcome such hurdles.

Zero-Knowledge Proofs

ZKPs allow the verification of a value without disclosing the value to the Verifier Smart (2016). ZKPs especially enable the integration of the minimisation property of SSI-IG. Broadly speaking, there exist two types of Zero-Knowledge Proofs: (1) exact proofs and (2) range proofs, both of which can have interactive or non-interactive variants. We propose the usage of ZKPs for their added benefits of non-disclosure and range proofs. For regular static values exact proofs should be used, whilst any form of attestation requiring a number, range proof should be used.

Attestation Design

We propose a design based on the work set out by Stokkink et al. (2020). The attestation procedure is visible in Figure 5. The design uses multiple phases, with optional steps. We make the distinction between two types of attestations [**TODO:** Rename "value-attestation request" to proof-request?]:

1. *Value-Attestation*: this type of attestation can be said to be the core type. It is responsible for incorporating a specific value into a Zero-Knowledge Proof. The verifiable-nature of attestations stems from this type. As visible in **??**, the design of this attestation allows for multiple *proof formats*, allowing for flexible selection of ZKPs and, thus, attestations. This disallows the lock-in of specific proof types, as any client can propose the usage of any type of proof, which can be used as long as the corresponding Authority supports the proposed type as well.

2. *Credential-Attestation*: this type of attestation is a reference to a Value-Attestation. This secondary type of attestation allows for the subsequent attesting of values, through attestation chaining: subsequent authorities can attest for the same value by attesting to the Value-Attestation as opposed to requiring a separate Value-Attestation. Credential-Attestations also refer to metadata, which allow for validity terms and sign dates.

There exist several benefits to this construction. Firstly, the aforementioned chaining of attestations allows for multiple authorities to attest to a value. As such, real-life signature scenarios can be modelled through attestations. This allows for concepts such as *segregation of duties* and other shared responsibility scenarios, in which multiple parties must attest for a certain claim in order to be valid. For instance, a credential attesting for the ownership of a driving license, may require a signature by both a government body handing motor vehicles and a local government. The ability for multiple attestations for a single value can prove to be capable of handling such real-life scenario's. Secondly, subsequent attestation do not require the knowledge of the plaintext value. For instance, continuing on the driving license example, a local government does not require extensive knowledge on the license itself, a signature by the responsible government body should be enough for them to attest. As a consequence, this aids in data minimisation on subsequent Authorities. Finally, in case of attestation properties such as validity terms, a renewal of an attestation can simply be a new Credential-Attestation for the Value-Attestation, not requiring the re-attestation for the actual data. Again, this aids in data minimisation and privacy, as the plain text values do not have to be disclosed. Additionally, different Authorities can adhere to different metadata of the same attestation without influencing other parties. By allowing different Credential-Attestations for the same Value-Attestation, different metadata is enabled to exist for the same Value-Attestation. For instance, different Authorities can set different expiration dates on the same Value-Attestation. Again, when the expiration date has passed, the issuing Authority can simply re-attest for the same Value-Attestation, generating a new signature for the Credential-Attestation.

Attestation Flow

The attestation flow consists of two phases, the Proof-phase and the Credential-phase which do not always require concurrent execution. More specifically, for a single to be attested claim, the Proof-phase requires a single execution, which must occur before the Attestation-phase. Whilst

Figure 1

Token Chain

subsequently, the Credential-phase can be performed indefinitely.

Proof-phase
The Proof-phase is initiated by a Subject. A Subject aims to have a claim attested to by an Authority. It does so by requesting an Attestation from the Authority. In this request, the Subject must make the attribute name, the to be used proof format, his public key apparent. This public key, is a one time used public key, of which the private key must be stored by the Subject. The usage of single used public/private key pairs, allows for additional privacy properties imposed on the system, which will be explained in **[TODO:** Add reference + write small section on this subject]. Additionally, any other information can be sent along, for instance the requested value. Note that the value is, thus, not required to be sent by the Subject. The implication of this, is that an Attestation can be made for the Subject, without the Subject knowing the exact value. This, hence, allows for the secure storage of information, in the form of a ZKP attestation on a client, without the actual revealment of the underlying value. The proof format allows for the proposition of different types of ZKPs used.

The receiving Authority can respond to the requesting, making him an issuing Authority. The Authority generates a Value-Attestation of the type defined by the *proof format*. This attestation, thus, incorporates the value belonging to the requesting attribute name. This attestation is sent back to the requesting Subject. After having received the Value-Attestation, the requesting Subject moves onto the *Attestation-phase*.

Credential-phase
In the Credential-phase, a requesting Subject requests an attestation for a certain Value-Attestation, making it a Credential. It does so by disclosing all already attested Credential-Attestations belonging to the Credential. The core of each Credential is an Attestation Token. Each Token contains the hash of a Value-Attestation and points to the previous Token. This has been visualised in Figure 1. The first token, comparable to a genesis-block in Blockchain structures such as that by **?**, contains the hash of the public key belonging to the Subject. Any subsequent Credential, thus, generated

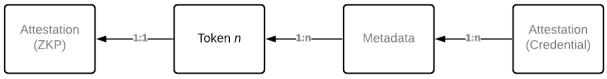

Figure 2

Data Structure Relationships

a new Token, occupying a place as a shackle in the chain. Therefore, when an Authority is requested to attest to a Credential, it request every previous token and, thus, the hashes of each previous attestation. After which, it can verify these attestations. As such, it is improbable for a client to attempt to hide the existence of an attestation or attempt to cheat the system, as otherwise the attestations of other Authorities become invalid (as the hash of the token will no longer be correct). Hence, as visible in the second phase of in Figure 5, after having received a Credential request, the Authority can possibly request any missing tokens until he gains confidence to attest for the Credential, creating a Credential-Attestation. Note that these Tokens do not reveal any information about the underlying Value-Attestation, as they merely contain the hash value. When an Authority attests to a Credential, it generates a signature for the hash of the corresponding metadata, which in turn points to a Token. This structure of referencing data structures is visualised in Figure 2. As is visible, a Token refers to a single Value-Attestation. However, multiple metadata instances may reference a single Token and, similarly, multiple Credential-Attestation are may reference a single metadata instance. These relationships allow for aforementioned properties and scenarios. As becomes apparent from this description, the second phase, i.e., the Credential-Phase, can thus be repeated indefinitely as numerous Authorities can co-attest for an Attestation.

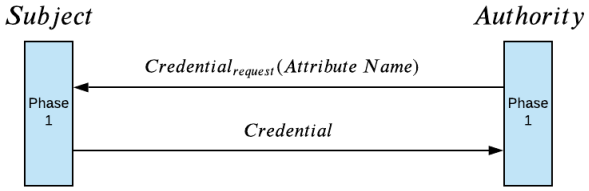

Figure 3

Attestation Presentation

Presentation

In order to verify attestation values, a presentation procedure must exist. As clients may decide themselves whether to share attributes, we propose the structure as visible in Figure 3 **[TODO:** Add token requests]. In this structure, an

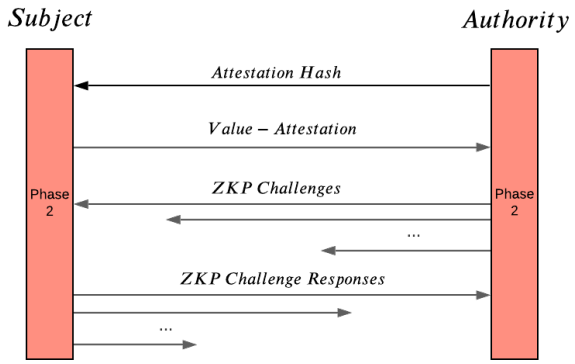

Figure 4

Interactive Verification

Authority requests an attribute with a specific name. A Subject can subsequently decide whether to respond to such a request and to disclosure the corresponding attribute. Note here that the credential request is not necessarily required, as a client can disclosure an attribute directly. However, the specification of a specifically named attribute, makes selective disclosure more transparent. Whilst, additionally, allowing the Authority to determine whether a specific credential is solicited. After a credential has been disclosed and, thus, presented, the Authority can verify its validity.

Verification

We propose two types of verification. Firstly, an interactive variant and, secondly, a fully non-interactive variant, enabling offline verification. The general flow of the interactive variant is visible in Figure 4. For active verification, the Authority requests the underlying Value-Attestation by sending the attestation hash to the Subject. The Subject may then consent through sending the requested Attestation. After the Authority receives the ZKP commitment, the Authority may send challenges to verify the underlying value. Note that for this to happen, the Authority must either be already aware of the value belonging to the attribute or the plaintext value must be shared. The sharing of the plaintext value can be done during presentation-time. This should be performed using encryption in order to preserve privacy, for instance through the use of RSA by Rivest, Shamir, and Adleman (1978). The second method for verification uses the attestations made by other authorities. In order for this attestation to pass, the list of attestors must contain an authority that is trusted by the verifying Authority. If this is the case, a Verifier may accept the value proposed by the Subject in case the metadata contains the hash of this value and the signature made by one of the acknowledged authorities over the metadata is valid. This approach does not require any connectivity between the Subject and Verifier, apart from the presentation itself. However, a presentation does not necessarily require any form of digital communication. It is, however, to note that this offline verification, thus, does not rely on any additional token requests and, as such, all tokens must either be made directly apparent to the Verifier during presentation-time or the verifier must make its decision based on the presented Attestation and his reliance on and knowledge of acknowledged authorities.

Revocation

Revocation is one of the main issues in Self-Sovereign Identity. As in real life contracts and other agreements may become invalid before their termination date, the ability to revoke attestation in SSI must be available as well. Several motivations exist for revocation:

- Erroneously signed data: in case data was signed accidentally.
- A Legally invalid contract: in case at a later instance it became apparent that the signed data can not be legally upheld.
- Premature termination of a contract: in case a certain breach of contract occurs.

Note that expiration is not one of these listed motivations, as time-bound attestations can be realised using signed metadata. It is important that revocation can never occur due to expiration, as some claims should never be able to be revoked. For instance, it should not be possible for an authority to revoke a signature indicating someone is of legal age (unless in the rare instance that it was erroneously signed and can be publicly verified that this was, indeed, the case), as this fact can never become false.

As IG-SSI is built without specialised validation nodes, present in some blockchain-based protocol such as Zhou, Li, and Zhao (2019), there is no trivial non-interactive solution of revocation of signatures. The trivial solution is to actively query signees (i.e., the responsible authorities) and verify that they still attest for the signed information. There exist multiple problems with this solution. Firstly, this querying requires interactivity with the signee(s) of an attestations. Whilst interactivity is not a problem per se, it does introduce additional overhead. Firstly, it requires the signee(s) to be online. Whilst availability often is a key characteristic in distributed systems, there is no guarantee that specific clients, i.e. the signees, are available. Secondly, this interactivity generates additional overhead in the verification process. Apart from challenging the presenting client, the signees have to be actively queried, introducing additional verification time and network traffic. Secondly, as a requirement for enabling this interactivity, a (network) connection to

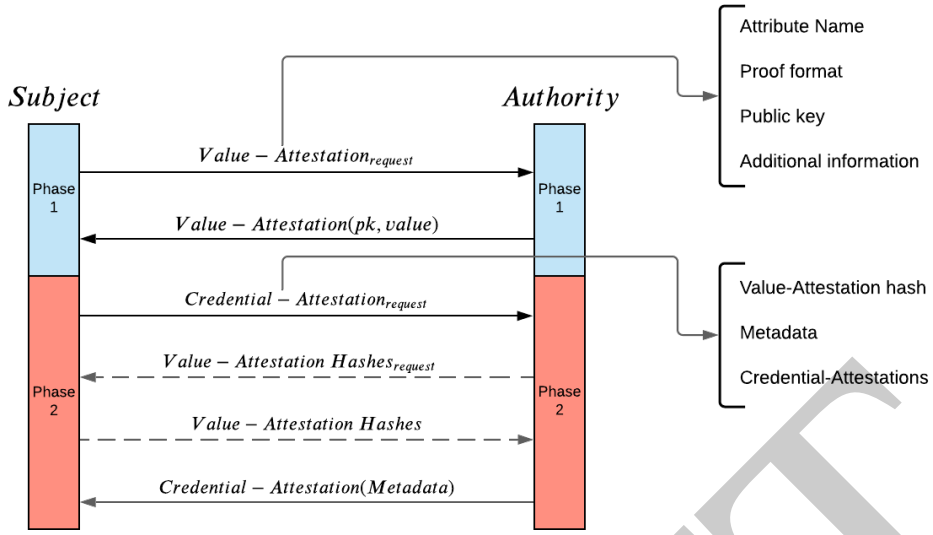

Figure 5

Attestation Flow

the signees must be available. This completely nullifies the possibility for offline verification. Next, we discuss our solution for revocation: *The Hybrid Revocation Model* (HRM). This model requires no additional interactivity during verification and enables offline-verification.

Hybrid Revocation Model

The Hybrid Revocation Model attempts to overcome the hurdle of interactivity whilst allowing for flexibility, enabling offline-verification. IG-SSI is fully distributed and as such, each node is equal. As a consequence, the client performing the verification must be aware of any revocations belonging to a presented attestation. Selecting specific nodes for distributing and holding revocation, would deteriorate the equality principle. As these nodes would, then, possess the ability to hide certain revocations from the network or from certain peers or could lead to collusion (Khovratovich & Law, n.d.). As such, revocations should be public data. I.e., every revocation should be visible to and known by every client.

The *hybrid* nature of the model, stems for its offline capabilities: during verification-time, clients do not require to be online. They merely require occasional synchronisation of revoked attestations through communication with other peers.

In HRM, each peer has the possibility to posses the same information about revocations. Revocations are propagated through the network, enabling each peer to store revocations from clients they trust. This concept builds upon the notion of Trusted Authorities. The general flow of the design can be seen in Figure 6. The protocol has three key concepts:

1. Trusted Authorities (TAs)
2. Propagation
3. Offline Revocation List (ORL)

Next, we explain each concept.

Trusted Authorities

In a fully distributed setting, client are responsible for each of their own actions. Meaning that revocation are as meaningless or meaningful as the extend to which they are used by the clients. This property makes it that clients themselves are able to acknowledge or rejects. revocations. A criterion on which a client is able to determine the validity of a revocation, is whether the Revoking Authority is trusted by the client. This is where we introduce the notion of Trusted Authorities (TAs). As mirrored by real life, a person has (relatively speaking) a choice whether to acknowledge a certain authority. With SSI aiming to be a digital extension to one's identity, one should also be able to make such an acknowledgement in the digital domain. As an added benefit, identification in the digital domain can prove to be more verifiable than physical verification. We propose the usage of a Trusted Authority Storage (TAS). In the TAS, the public key and the public key hash of a TA are stored. We make the distinction between acknowledged and Unacknowledged Authorities (UAs), where Acknowledged Authorities are a TAs. As discussed previously, client roles are neither static nor mutually exclusive. As a consequence,

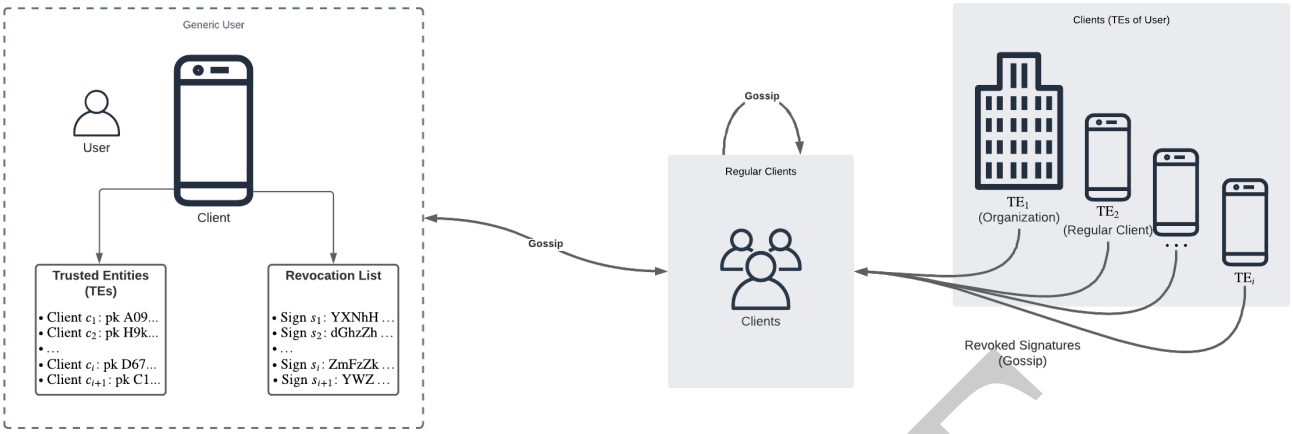

Figure 6

The Hybrid Revocation Model (HRM)

potentially every client can be an Authority. However, it is up to a client to determine whether an authority is trusted and, hence, acknowledged. In terms of distributed revocation: a client aims to accept only those revocations of which he knows that he can trust the authority. The results of acceptance are the storage of the revoked signatures and propagation towards network.

Propagation

In order to safeguard availability in the network and enable offline verification, we propose the propagation of revocations throughout the network. This requires two means: firstly, a verifiable revocation format and, secondly, a propagation protocol for the revocations. We propose the structure as visible in Table 1. This design, in addition to the revoked hashes, includes a public key hash, a version number, a specification for the used hashing algorithm and a signature. The public key hash allows for the retrieval of the public key in case said key belongs to a TA acknowledged by the receiving client. This public key can, thus, be retrieved by querying the TAS. In case the public key belongs to a TA, the signature can be verified by concatenating the version number with the revocations. Unique version numbers allow clients to ensure that they are either fully synced with the network or are missing certain revocation versions. The revocations themselves are to be the hashes belonging to the attestation metadata. This, thus, invalidates any attestations made to this metadata and the token it points to. As a benefit, this reduces overhead when presenting attestations, as solely based on the metadata, an attestation can be deemed to be valid or revoked. The hashing algorithm specification improves the transparency and robustness of the schema. For instance, hashing algorithm recommendation may differ in the future due to e.g. efficient collision finding. Allowing for specification enables the interchanging of this algorithm, aiding future-proofness and flexibility.

The propagation itself requires a protocol that ensures information is (eventually) spread across the entire network, whilst also ensuring that unavailable nodes receive the information at a later instance. For this, we propose the usage of gossip protocols with interval re-transmission. Gossip protocols are communication protocols which allow for the periodic exchange of data with (random) peers (Kwiatkowska, Norman, & Parker, n.d.). The periodic exchange of data with peers, makes gossip protocols a prime candidate for the realisation of distributed revocation. Furthermore, in order to decrease the overhead of gossiping a theoretically unbound number of signatures, we propose the usage of a multi-step update procedure. This procedure has been visualised in **??**. This precedure is split-up in two phases. Firstly, a gossiping client gives notice to a client that it possesses specific authority-version pairs, containing the public key hash of an authority and the latest version it is aware of. Next, the receiving client can request an update by sending back the versions of the TAs that they are unaware of. After this selective update request, the gossiper can send the request updates. This extra step of selective requesting relieves a large amount of data as clients are not necessarily either be not interested in revocations by certain authorities as they may be considered UAs or a client may already be fully synced.

Offline Revocation List

Any valid received revocation should be stored by a client for later reference. The storage of revocations allow for offline (in)validation of attestations. This storage we deem the Offline Revocation List (ORL). Whilst no specific storage structure is required, we do propose the usage of bloom fil-

Table 1

Verifiable Revocation Update Format

Authority key hash	5e2bf57d3f40c4b6df6...
Version	1701
Hashing Algorithm	SHA3-256
Signature	422c06fbb4fbd23d33...
Revocations	b788c5b28dba2fc6a0... 7f2519609cf157d7e9... ... e2d7610dcb53724675...

ters for member checking. A Bloom filter is a memory- and time-efficient probabilistic data structure, which allow for efficient membership operations (Bloom, 1970). As yearly up to 340.000 identity documents are stolen in a country as The Netherlands, the same amount of revocations must be possible on a year basis (Nieuwsuur, 2019). As such, revocation membership checking can prove to become quite expensive both memory- and runtime-wise. Even with the most efficient algorithms such as Binary search, with a runtime complexity of $O(\log(n)$, such a search can be too long, usability-wise. As such, we propose the usage of membership verification through Bloom filters, after which a membership search on the actual data is only performed in case of a possible match. Additionally, it can be said that the probability of encountering a revoked attestation should be extremely unlikely. As we assume the majority of the nodes to be honest, they have no incentive to attempt to cheat the system. As such, Bloom filters with their property of ensuring an item has no membership in case the filter does not contain it and, thus, only having to validate using the actual data in case the filter may contain the item, Bloom filter can prove to achieve much stricter execution timings for validation.

Results

For the realisation of IG-SSI, we implemented three semantic layers, namely:

1. **Attestation Layer**: abstracts the signing of Zero-Knowledge Proofs and verification.

2. **Credential Layer**: abstracts the attestations of Authorities over ZKPs and enables chaining of attestations.

3. **Revocation Layer**: abstracts the handling of revocations over credentials.

These three layers are built on top of the academic communication protocol of IPv8[3] primarily based on the works by Halkes and Pouwelse (2011); Zeilemaker et al. (2013). The selection of IPv8 stems from firstly its academic background, proving its viability through various publications. Secondly, IPv8 allows for direct client-to-client communication, hence, enabling a fully distributed infrastructure at the core of the solution. Finally, IPv8 does not require (expensive) Proof-of-Work algorithms utilised by Blockchain structures such as Nakamoto (2009) and Buterin (2013). In addition to the aforementioned semantic layer, a secure multi-party communication channel has been developed.

The Semantic Layers

Next, we discuss the implementation of each of the semantic layers.

Attestation Layer

The *attestation*-layer abstract the logic for signing and verifying the ZKPs used. Whilst not necessarily requiring ZKPs, as the design allows specification and, thus, negotiation of used proof formats, ZKPs are highly recommended due to the intrinsic properties they introduce to the system. The design itself is, thus, proof-agnostic as one can implement any type of proof. Per choice, two types of ZKPs are implemented. Firstly, a ZKP proof allowing arbitrary data and the verification of exact values. For this, the algorithm proposed by Boneh, Goh, and Nissim (2005), allowing verifiable computation through 2-DNF formulae over bits. Boneh et al. (2005) is a homomorphic public-key encryption scheme with, as proven by Boneh et al. allows for universally verifiable computation, a property which is desirable in Self-Sovereign Identity. Additionally, this allows for interoperability with the schema proposed by Stokkink et al. (2020). Secondly, the range ZKP proposed by Peng and Bao (2010) has been implemented. This ZKP allows for the encoding of integer values laying in a specific range. Peng and Bao (2010) requires constant costs, proving to be more efficient than previously proposed solutions. We implemented the commitment scheme proposed by **?** in order to realise the range proof by Peng and Bao (2010). Both of these proofs are interactive. However, as shown by Koens, Ramaekers, and Van Wijk (2018), the schema introduced by Peng and Bao (2010) can be made non-interactive.

Revocation

For revocation, we implemented a custom gossip protocol

For the ORL, a Bloom Filter (Bloom, 1970) has been implemented for memory-usage and run-time improvements. Based on the expected 300.000 lost identification documents per year, as presented by Nieuwsuur (2019), the following memory and time considerations can be made. Firstly, a storage for 300.000 hashes of 32 bytes each, results in a space usage of at least 9.2 megabytes. Whilst a Bloom fil-

[3]For the official (Python) documentation of IPv8, see `https://py-ipv8.readthedocs.io/en/latest/`

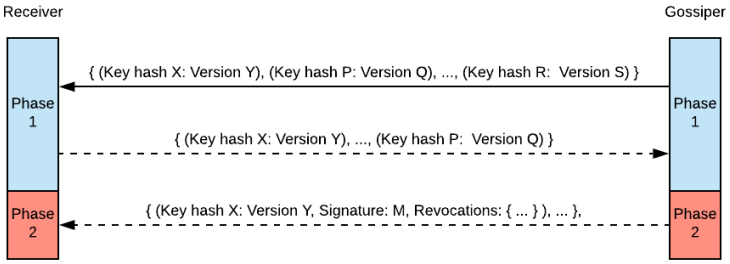

Figure 7

Multi-step Update Procedure

ter with a probability of a false positive of 1 in 100 million and 27 hashing functions, can achieve such a storage requiring merely 1.43 megabytes of storage. Whilst both such space requirements are easily satisfied by modern handheld devices (GSMArena, 2018), the run-time benefits do introduce a noteworthy improvement. **??** showcases the speed-up provided by Bloom filters. The Bloom filter in questions uses the following parameters: **[TODO:** add params]. On a dataset of 100,000 revoked hashes, one can see that, as expected, the runtimes increase linearly. The x-axis varies the percentage of the candidate which is an actual member of the test data set. In other words, the percentage of actual matches increases in each subsequent measure. As expected, the verification utilising solely a Bloom filter is not impacted by this variation. Similarly, verification solely utilising Binary Search is also relatively unimpacted. The variation only utilising binary search on a possible match in the Bloom filter, is impacted the most. This variant only makes (expensive) I/O operations when the Bloom filter reports a possible match. As becomes apparent, the benefits from the Bloom filter decrease with the increase of the membership percentage. Hence, the speed-up is most prominent with lower membership percentage. In terms of attestation verification, a Bloom filter is thus most beneficent in case the vast majority of the encountered attestation are non-revoked and, thus, valid. We draw the conclusion based on the reported statistic by Nieuwsuur (2019), which stated that in 2018, in the Netherlands nearly 340.000 official identification documentation was lost. Percentage-wise, this leads to an annual 2% loss based on the 17.18 million residents of that year CBS (n.d.). Note that this estimation does not include the number of different identification documents hold by a resident (e.g. driving license, passport, and identification card), as a consequence this actual number will most likely differs greatly. Also note that the properties of physical identification measures do not do no directly translate to any digital variants, as, most desirable, digital credentials are far more difficult to lose. However, this showcases that it can be expected to

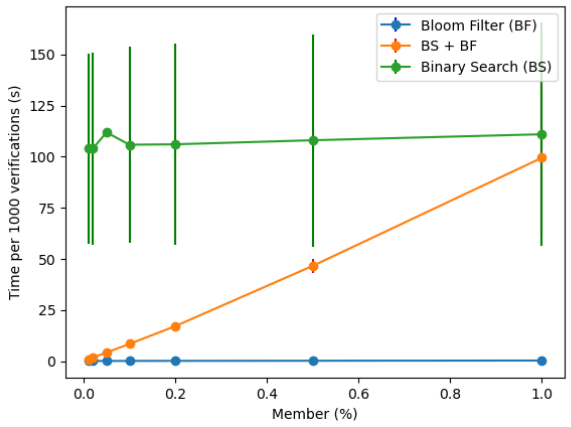

Figure 8

Verification runtime per 1000 transactions (n=100,000

encounter far more valid attestations than revoked ones. Especially with the assumption that the majority of the network is honest. To conclude, we deem the speed-up benefits provided by the usage of Bloom filters to be significant.

References

Bloom, B. H. (1970, 7). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*(7), 422–426. Retrieved from `https://dl.acm.org/doi/abs/10.1145/362686.362692` doi: 10.1145/362686.362692

Boneh, D., Goh, E. J., & Nissim, K. (2005). Evaluating 2-DNF formulas on ciphertexts. In *Lecture notes in computer science* (Vol. 3378, pp. 325–341). Springer Verlag. Retrieved from `https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-540-30576-7_18` doi: 10.1007/978-3-540-30576-7{_}18

Buterin, V. (2013). A next generation smart contract & decentralized application platform. *Ethereum Foundation*.

Cameron, K. (2005). The laws of identity. *Microsoft Corp*, *5*, 8–11.

CBS. (n.d.). *Bevolkingsteller.* Retrieved from https://www.cbs.nl/nl-nl/visualisaties/dashboard-bevolking/bevolkingsteller

GSMArena. (2018, 4). *Counterclockwise: RAM capacity through the years* . Retrieved from https://www.gsmarena.com/counterclockwise_ram_capacity_through_the_years-news-30756.php

Halkes, G., & Pouwelse, J. (2011). UDP NAT and firewall puncturing in the wild. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 6641 LNCS, pp. 1–12). Springer, Berlin, Heidelberg. Retrieved from https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-642-20798-3_1 doi: 10.1007/978-3-642-20798-3{_}1

Khovratovich, D., & Law, J. (n.d.). *Sovrin: digital identities in the blockchain era* (Tech. Rep.). Retrieved from http://www.credentica.com/the

Koens, T., Ramaekers, C., & Van Wijk, C. (2018). *Efficient Zero-Knowledge Range Proofs in Ethereum* (Tech. Rep.). ING.

Kwiatkowska, M., Norman, G., & Parker, D. (n.d.). *Analysis of a Gossip Protocol in PRISM* (Tech. Rep.). Retrieved from http://www.prismmodelchecker.org/casestudies/gossip.php

Nakamoto, S. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System* (Tech. Rep.). Retrieved from www.bitcoin.org

Nieuwsuur. (2019, 7). *We verliezen massaal onze paspoorten: fraudemeldingen verdubbeld | Nieuwsuur.* Retrieved from https://nos.nl/nieuwsuur/artikel/2292728-we-verliezen-massaal-onze-paspoorten-fraudemeldingen-verdubbeld

Peng, K., & Bao, F. (2010). An efficient range proof scheme. In *Proceedings - socialcom 2010: 2nd ieee international conference on social computing, passat 2010: 2nd ieee international conference on privacy, security, risk and trust* (pp. 826–833). doi: 10.1109/SocialCom.2010.125

Rivest, R. L., Shamir, A., & Adleman, L. (1978, 2). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126. Retrieved from https://dl-acm-org.tudelft.idm.oclc.org/doi/abs/10.1145/359340.359342 doi: 10.1145/359340.359342

Smart, N. P. (2016). In *Cryptography made simple* (p. 425–438). Springer.

Stigler, G. J. (1964). A theory of oligopoly. *Journal of Political Economy*, *72*(1), 44–61. Retrieved from http://www.jstor.org/stable/1828791

Stokkink, Q., Epema, D., & Pouwelse, J. (2020). A Truly Self-Sovereign Identity System. *arXiv preprint arXiv:2007.00415*.

Stokkink, Q., & Pouwelse, J. (2018). Deployment of a blockchain-based self-sovereign identity. In *2018 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)* (pp. 1336–1342).

Zeilemaker, N., Schoon, B., & Pouwelse, J. (2013). *Dispersy bundle synchronization* (Tech. Rep. No. PDS-2013-002). TU Delft. Retrieved from http://www.pds.ewi.tudelft.nl/fileadmin/pds/reports/2013/PDS-2013-002.pdf

Zhou, T., Li, X., & Zhao, H. (2019). EverSSDI: Blockchain-based framework for verification, authorisation and recovery of self-sovereign identity using smart contracts. *International Journal of Computer Applications in Technology*, *60*(3), 281–295. doi: 10.1504/IJCAT.2019.100300