SSI-AWARE: Self-sovereign Identity Authenticated Backup with Auditing by Remote Entities

Philipp Jakubeit, Albert Dercksen, Andreas Peter

SSI-AWARE: Self-Sovereign Identity Authenticated backup With Auditing by Remote Entities

Philipp Jakubeit[1,2], Albert Dercksen[2], and Andreas Peter[1]

[1] Services and Cybersecurity Group, University of Twente, Enschede, The Netherlands
<P.Jakubeit,A.Peter>@utwente.nl
[2] Nedap N.V., 7141 DC Groenlo, The Netherlands
<Wouter.Kuijper,Albert.Dercksen>@nedap.com

Abstract. The self-sovereign identity (SSI) model entails the full responsibility and sovereignty of a user regarding his identity data. This identity data can contain private data which is solely known to the user. The user himself is therefore required to manage the whole lifecycle of his private data, including the backup and restore. We show that prior work on how to backup and restore the user's identity data does not meet the requirements of the SSI setting, and we present the first solution which does meet the requirements. Authenticated backup with auditing by remote entities (AWARE) combines SSI sustaining aspects and extends them to create a truly self-sovereign backup-and-restore protocol. In AWARE, trusted, physically met humans, called custodians, hold a secure device. Custodians with a secure device offer an offline backup possibility and a secure channel. The backup and restore are audited by commits on a publicly accessible distributed ledger. These commits are answered by auditing services which are required during restore. Only some auditing services hold relevant data for a restore. The self sovereignty of the user lies in the exclusive information which auditing services hold relevant data. AWARE is the first backup-and-restore mechanism that fully complies with the SSI model. We perform an in-depth security-risk analysis of AWARE, showing a risk rating which is comparable to the best risk rating of related non-SSI-compliant backup-and-restore mechanisms. We instantiate the AWARE protocol with cryptographic primitives providing a high security level of 256-bit. We show its implementation feasibility by providing a simulation of AWARE, and conclude with an estimated performance analysis on a microcontoller architecture based on our simulation and implementation results in the literature.

1 Introduction

The amount of digital information is ever-increasing. The International Data Corporation predicts an annual growth rate of 61% from 2018 to 2025 to the

global datasphere's volume of 175ZB [9]. With the majority of this data being user generated. From 2010 to 2019 the amount of internet users doubled and nearly reached 60% of the world population [23]. The user generated data consists of content and identity data. This identity data is traditionally known to the user and the service. The identity data can include private data, which is a privacy challenge for the user and an increasing challenge for the services. The General Data Protection Regulation (GDPR) [12], an EU law regarding data protection and privacy of the EU citizens, exemplifies the legislative relevance of privacy.

Several identity models have been proposed that achieve different levels of privacy protection [1]. In case of the *isolated identity model* the service, a trusted authority (TA) stores and manages the user's identity data directly. In case of the *central identity model* an identity provider, a TA, stores and manages the user's identity data and transfers it to a service. In case of the *federated identity model* the identity data is stored and managed distributed across multiple identity providers (TAs). In case of the *user-centric identity model*, a service manages the identity data, but a device is distributed to the user, a security token (ST). It empowers the user to store his identity data. It can be realized by a smart card or other dedicated hardware. The sharing of identity data requires explicit user consent, but the identity data is still managed by an identity provider. The *self-sovereign identity* (SSI) model emerged due to the rise of distributed ledger (DL) technology, it extends the user-centric identity model. In the SSI model, the user stores his own identity data and also manages his identity data over its entire lifecycle. Therefore, the user instead of a central authority, is in control of the identity data and the user decides how personal attributes and credentials are shared [4].

An essential but overlooked aspect of SSI is backup and restore of the *private data* PD_u of a user u. The user is responsible for the entire lifecycle of his private data, therefore, also for backup and restore. Private data PD_u is per assumption not known to any other entity than the user u. Therefore, the user u is the only entity capable of conducting the backup of PD_u. The process of backup is to store a copy of the data such that it is available for restore in case of lost access. The process of restore is to access the backed up copy of the data. Abilities gained by the private data are given to any entity who can access it. The access to the backed up data thus determines who is sovereign. Access control consists of three steps: authentication, authorization, and auditing [26]. Authentication describes that the authenticity of an entity trying to access the data must be verified. Authorization describes the decision whether to grant or deny access to the data for a specific entity. Auditing describes the capability of the system to trace the actions of its participants. Since sovereignty lies in the access to the data, we define:

> *A self-sovereign solution must require the user to conduct the step of authorization and the user must also choose who authenticates and who audits the restore.*

We show that in prior works on how backup and restore can be conducted the user is not sovereign in case of lost access to his private data. We therefore

present our self-sovereign backup-and-restore protocol of authenticated backup with auditing by remote entities (AWARE). The user is empowered to decide if the restore is granted. We achieve this by backing up the data to physically met custodians who are audited on a public DL. The user is just required to hold a low entropy information (about 6 digits) to restore his PD_u. The AWARE protocol further extends the STs of the user-centric model by interconnectable security tokens (IST's). An IST is an ST which has a procedure to exchange specific information with other IST's. We conduct a security-risk analysis of the AWARE protocol and compare the results to the current proposals. We show that the AWARE protocol is the only backup-and-restore mechanism providing self sovereignty while its worst risk rating is close to the best risk rating of the proposals not offering self sovereignty.

2 The Problem of Backup and Restore in the SSI model

The problem of backup and restore in the SSI model is that the user u has private data PD_u which is solely known to him. He wants to backup PD_u such that after he lost access to PD_u he can regain access to it. A perfect requirement would be that only the user u can access PD_u while any other entity $e \neq u$ cannot. However, over an insecure channel perfect authenticity cannot be achieved [21]. Realizable requirements are:

1. The user u has a high likelihood of accessing his backed up PD_u.
2. Any other entity $e \neq u$ has a very low likelihood of accessing the backed up PD_u.

Two systems realizing SSI management explicitly offer a suggestion on how to conduct backup and restore: Sovrin [30] and uPort [7]. Sovrin proposes the use of distributed storage devices or social backup, backing up at entities the user trusts (custodians). uPort proposes a smart contract based solution in which the private key can be swapped to another key, based on the rules specified in a smart contract and promotions of earlier specified online entities the user trusts.

By definition, a backup-and-restore mechanism is self sovereign if the user performs the authorization during restore and chooses who authenticates and who audits the restore process. The authentication must be performed by some other entity than the user. Since the user is not capable of authenticating himself after data loss. Neither the user nor the authenticating entity can audit themselves. Therefore, the auditing must be conducted by another entity than the user or the authenticating entity. The audit trail must be accessible to the unauthenticated user as he is not capable of accessing his private data. Next to the steps of access control, the accessibility itself must be minimized. Therefore, the backed up private data should be stored offline.

In the literature we identified four backup-and-restore mechanisms: *Trusted authorities*, one identity service knows all the private data; *Local backup*, a user backs up his private data on several devices, which the user distributes (e.g. at home, deposit box, at work); *Social backup*, a user distributes his private data across several personally trusted entities (custodians); *Smart contract* based,

the user allows trusted online entities to promote a new key into his controlling smart contract. A smart contract is a protocol executed on a DL. It empowers its participants to conduct credible transactions without the need for third parties. In Table 1 we show per backup-and-restore mechanism which entity conducts which step of access control and whether the backup is accessible online or offline.

B-and-R mech.	Authentication	Authorization	Audit	Accessibility
Trusted authority	TA	TA	Not defined	Online
Local backup	Not defined	Not defined	Not defined	Not defined
Social backup	Custodians	Custodians	Not defined	Offline
Smart contract	Trusted online entities	Smart contract	DL	Online

Table 1. Per backup-and-restore mechanism the table presents which step of access control is conducted by which entity as well as the accessibility of the private data.

In case of the *TA*s every step of access control is managed by a TA. Authentication is only possible via a public channel (the Internet) and the data is stored online. It is thus not self sovereign. In case of the *local backup* the steps of access control and the accessibility are not clearly defined. It is therefore not self sovereign. In case of *social backup* authentication is conducted by custodians via a secure channel. The authorization, however, is also conducted by the custodians without any defined procedure of audit. It is thus not self sovereign. In case of the *smart contract* based backup-and-restore mechanism the authentication is conducted by trusted entities via a public channel which can trigger the authorization of the smart contract by promoting a new key. It is not self sovereign and the private data is stored online. However, it offers a publicly accessible auditing trail by writing the promotions on the DL. In conclusion, none of the backup-and-restore mechanisms in the literature is self sovereign. The aspect of a secure channel between the user and his custodians and a publicly accessible auditing trail on the DL, however, form a starting point for us to build a self sovereign solution.

3 AWARE

The self sovereignty of the user, even under data loss, is the aim of our protocol on authenticated backup with auditing by remote entities (AWARE). We have the following participants: *The user* performs the authorization and decides who conducts the authentication and who conducts the auditing. *Custodians* are entities the user trusts and meets physically, they conduct the authentication. Likely candidates are family and friends for personal data backup and colleagues for company related data backup. *Auditing services* conduct the auditing, logging on a DL. The accessibility of the backed up PD_u is offline to decrease the attack surface. The physically met custodians in combination with their IST's allow for offline data storage and a secure channel. In our AWARE protocol only auditing requires online communication.

Notation All participants of the protocol are modeled as elements of the set of entities E. We abbreviate a specific entity by a small letter (e.g. the user $u \in E$).

We indicate the relation to a specific entity by placing this entity in the index (e.g. the user u's private data is abbreviated as PD_u). If an entity belongs to a subgroup of known size we denote it with a numbering in the index. The i'th user of the subgroup of n users U is denoted with $u_i \in U$ for $i \in \{1, \ldots, n\}$.

Regarding functionality, we use the following notation: We denote the symmetric-key encryption of data D by a symmetric-key k with $Enc_k(D)$ and respectively the decryption such that $Dec_k(Enc_k(D)) = D$. Asymmetric-key encryption of data D by an asymmetric-key pek with $Enc_{pek}(D)$ and respectively the decryption such that $Dec_{sek}(Enc_{pek}(D)) = D$. We denote the splitting of data D into n shares by secret sharing $D \xrightarrow{SS_{(t,n)}} \{s_1(D), \ldots, s_n(D)\}$ such that t of these shares can be used to reconstruct the data D. Regarding some set B we denote a uniformly random pick with $a \xleftarrow{\$} B$ for $a \in B$.

Parameters In Table 2 the parameters of the AWARE protocol description are presented in their contexts.

Variable	Context
u	The user.
C_u	The custodians of user u.
$\mathcal{AS}_u$	The auditing service list of user u.
$\mathcal{ASS}_u$	The auditing services shares from the user u.
π_u	The policy of user u.
$l \in \mathbb{N}$	The key length.
k	The key $k \xleftarrow{\$} \{0,1\}^l$
n,t	n-out-of-t shares of $\mathrm{Enc}_k(PD_u)$ (using secret sharing).
r,q	r-out-of-q shares of k (using secret sharing).
x,y	x-out-of-y key promotions required in policy π_u
$p \geq q$	The total amount of auditing services.
h	The maximum of publishable restore releases.
ρ	Memorizable, low entropy information of q services holding a key share.

Table 2. Parameters of the AWARE protocol.

Assumptions A user u of our system,

- holds private data PD_u,
- is capable of memorizing the low entropy information ρ (q numbers less than p),
- possesses an interconnectable security token (IST_u).

The concept of an interconnectable security token (IST) is an extension on the ST used in the user-centric identity model. It stores PD_u securely, just accessible to the user u. The extension lies in the connectivity of the IST. While a smartcard or ordinary ST device can just connect to a smartphone or computer the ISTs are capable to interconnect. An IST_u of a user u guarantees that

- private data PD_u stored on IST_u is only accessible to the user u,
- it is offline by default and the user needs to specifically enable communication,

– it is interconnectable with other IST_e for $e \neq u$ via a short range channel (e.g. Infrared [3], Near Field Communication [16]),
– it is connectable via another channel to an internet capable device (e.g. smartphone, computer, etc.).

The participants of the AWARE protocol are the user u who has n custodians ($\mathcal{C}_u$) and p auditing services ($\mathcal{AS}_u$). The user u and his custodians $\mathcal{C}_u$ posses an interconnectable security token (IST). All participants hold two public-secret key pairs. A keypair for signing (psk,ssk) and a keypair for encryption (pek,sek). The user u further holds private data PD_u which is solely known to u and which should be backed up.

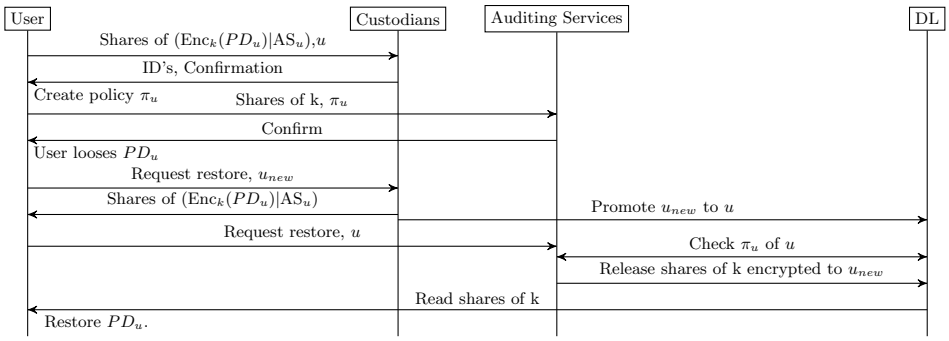

Fig. 1. Overview sequence diagram of AWARE.

Overview The AWARE protocol (Figure 1) consists of interaction from the user with his custodians and auditing services. The user chooses p auditing services and sends a share of his PD_u concatenated with the list of his auditing services and his public signing key to each of his n custodians. The user creates a policy π_u in which he specifies that x of y custodians are required to promote a new identity on the DL and the maximum amount of restore releases h. The user creates p auditing shares. He creates q key shares of the key k and adds $p - q$ fake shares. Subsequently, he distributes all auditing shares randomly over the p auditing services. The user receives ρ, the q auditing services holding a key share. In case of access loss to PD_u, the user approaches t custodians, requests his data and restores his encrypted PD_u and $\mathcal{AS}_u$. He approaches r auditing services from the ρ auditing services from $\mathcal{AS}_u$ and requests a restore. The auditing service checks the policy and only if it is met releases his share of k encrypted to the new identity of the user. The auditing service is aware of this new identity as it is promoted by the custodians on the DL. The user is then able to read the encrypted shares, decrypt them and reconstruct k. With the key k the user can decrypt his PD_u.

Initialization To be fully self sovereign, the user is required to choose the system parameters. He has to choose $n = |\mathcal{C}_u|$ the total amount of custodians, $t \leq n$ the amount of custodians required for a restore, $p = |\mathcal{AS}_u|$ the total amount of

auditing services, $q \leq p$ the total amount of auditing services holding a key share, $r \leq q$ the amount of auditing services required for a restore, and $k \xleftarrow{\$} \{0,1\}^l$ the key k of size l which must be generated uniformly random.

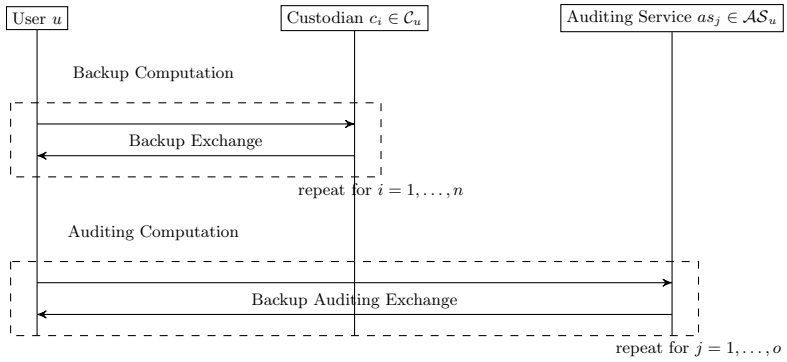

Fig. 2. Sequence diagram of our backup protocol.

Backup The basic structure of our backup protocol is illustrated in Figure 2. In the following we elaborate the details per step:

1. Backup Computation:
   - The user u encrypts his private data PD_u with the key k, $Enc_k(PD_u)$.
   - The user u appends $\mathcal{AS}_u$ and splits it $(Enc_k(PD_u)|\mathcal{AS}_u) \xRightarrow{ss_{(t,n)}} \{s_1(Enc_k(PD_u)|\mathcal{AS}_u), \ldots, s_n(Enc_k(PD_u)|\mathcal{AS}_u)\}$.
2. Backup Exchange, the user u meets with each custodian $c_i \in \mathcal{C}_u$ (for $i \in \{1, \ldots, n\}$):
   - The user u receives the public signing key psk_{c_i} of the custodian i.
   - The user u provides his public signing key psk_u and a share $s_i(Enc_k(PD_u)|\mathcal{AS}_u)$.
3. Auditing computation:
   - The user u chooses a policy, π_u.
     (a) psk_{c_i} for $i = 1, \ldots, x$ from which at least y must promote the user.
     (b) h, the maximum number allowed of published restore releases.
   - The user u splits his key k, $k \xRightarrow{ss_{(r,q)}} \{s_1(k), \ldots, s_q(k)\}$.
   - The user u generates $p-q$ random fake shares f and appends them to his list of auditing shares $\mathcal{ASS}_u$, $\mathcal{ASS}_u = \{s_1(k), \ldots, s_q(k), f_1, \ldots, f_{o-q}\}$.
   - The user u performs a truly random permutation on $\mathcal{AS}_u$ to receive $\text{Perm}(\mathcal{AS}_u)$. The q servers holding a key-share are outputted as ρ.
   - The user u encrypts each auditing share appended with his policy π_u to the corresponding service in the permuted auditing service list, $Enc_{pek_{y_j}}(a_j|\pi_u)$ for $a_j \in \mathcal{ASS}_u$, $y_j \in Perm(\mathcal{AS}_u)$ and $j \in \{1, \ldots, o\}$.
4. Backup Auditing Exchange:
   - The IST_u is connected to an online device and each $Enc_{pek_{y_j}}(a_j|\pi_u)$ signed with his psk_u in the order of $\mathcal{AS}_u$ is uploaded to as_j.
   - The auditing service as_j confirms the receipt.

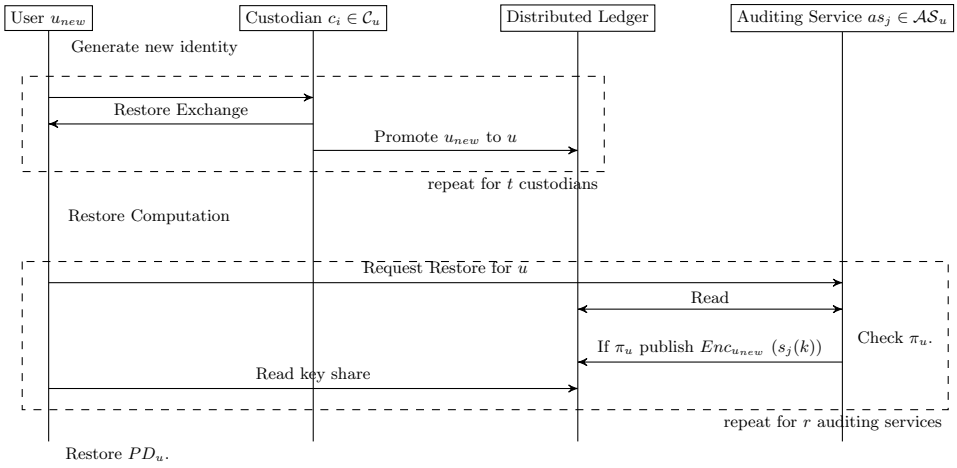

Fig. 3. Sequence diagram of our restore protocol.

Restore Assuming that the user u looses access to his PD_u he needs to engage in a restore protocol. For this we assume a new identity u_{new} of the user as his old identity is lost. The basic structure of our restore protocol is illustrated in Figure 3. In the following we elaborate the details per step:

1. Generate new identity:
   - The user generates a new key pairs ($psk_{u_{new}}$,$ssk_{u_{new}}$) and ($pek_{u_{new}}$,$sek_{u_{new}}$).
2. Restore Exchange:
   - The user u approaches a subset of t of his custodians and provides them with his new public encryption key $pek_{u_{new}}$.
   - The custodian c_i physically authenticates user u and provides him with his share, $s_i(Enc_k(PD_u)|\mathcal{AS}_u)$ for $i \in \{1, \ldots, n\}$.
3. Promotion:
   - Each of the t custodians promotes $pek_{u_{new}}$ for psk_u on the DL.
4. Restore Computation:
   - The user u reconstructs $Enc_k(PD_u)$ and $\mathcal{AS}_u$ from the t shares he received.
   - The user u approaches the auditing services holding a key share, by applying his low entropy information ρ.
5. Restore Auditing Exchange:
   - The user u request a key share from r auditing services.
   - If the auditing service $as_j \in \mathcal{AS}_u$ for $j \in \{1, \ldots, q\}$ receives a restore request for u, it checks the corresponding policy π_u and releases $Enc_{pek_{u_{new}}}$ $(s_j(k))$ if the policy π_u is met.
6. Restore:
   - The user u reads the encrypted key shares on the DL.
   - The user decrypts r key shares with his $sek_{u_{new}}$ and reconstruct k. He uses it to decrypt the encrypted private data $Dec_k(Enc_k(PD_u)) = PD_u$.

4 Security-risk Analysis

Security. The aim of AWARE is to allow the user u to conduct backup and restore self sovereignly. The authentication is conducted by carefully chosen custodians. The confidentiality and integrity of the data are maintained by choosing additional auditing services to audit the process of authentication and authorization. The AWARE protocol can be split into the sub-protocol conducted between the user and his custodians and the sub-protocol conducted between the user and his aware services. In our analysis we focus on both of the sub-protocols. For the protocol conducted between the user and his custodians it holds that if less than t custodians are compromised the private data's security is determined by the secret-sharing scheme used. For the protocol conducted between the user and his auditing services it holds that if less than r of the auditing services are compromised the key's security is also determined by the secret-sharing scheme used. If Shamir's secret sharing [27] is used the private data is information-theoretically secure if less than t custodians are compromised and the key is information-theoretically secure if less than r auditing services are compromised. Therefore, we assume for our analysis that t or more custodians are compromised and that r or more auditing services are compromised. We further assume that the custodians and auditing services do not collude and that ρ is only known to the user.

Integrity is the property of maintaining and assuring the accuracy and completeness of data over its entire lifecycle [17]. The custodians hold the shares of the private data encrypted with key k. The auditing services hold the shares of the key k. If t or more custodians are compromised they can restore the encrypted private data and can manipulate it, however, altering would render the encrypted private data undecryptable by k. Therefore, the accuracy and completeness of the private data is maintained and assured over its entire life cycle. If r or more compromised auditing services would manipulate the key k, the encrypted private data would be undecryptable and thus accuracy and completeness of the key k is also maintained and assured over its entire life cycle.

Confidentiality is the property that information is not made available or disclosed to unauthorized entities [17]. Our AWARE protocol makes use of secret sharing for the distribution of the encrypted private data PD_u to the custodians and the key k to the auditing services. If t or more custodians are compromised, the encrypted private data can be restored and is now protected by the security assumptions of the symmetric cipher used. To retrieve the key k an illegitimate restore attempt must be conducted. This attempt requires the compromised custodians to get the key k which will be logged on the DL. In this situation the user can engage. However, this is still insufficient to guarantee self sovereignty. The auditing services reply and proceed with the restore part of the AWARE protocol for a new identity, not distinguishing the user from an adversary. Therefore, we introduced auditing services holding fake shares and the maximum of potential restore releases h. The adversary is additionally required to choose r of the q auditing services holding a key share from the p auditing services in total. The chances to guess r auditing services from the q auditing services holding a

key share from p auditing services in total without knowing ρ is $\prod_{i=0}^{h-1} \frac{q-i}{p-i}$. If we choose the strictest policy that the amount of allowed restore releases is equal to the amount of key shares required, thus fixing $h = r$, the chance to guess r key share holding auditing services is $\frac{q!(p-r)!}{(q-r)!p!}$. If r or more auditing services are compromised we have the same chance that these r auditing services holding key shares. Without compromised custodians, however, the adversary gets no information about PD_u and cannot even verify if the illegitimately restored key k' equals the key k.

Availability is the property that information is accessible and usable on demand by an authorized entity [17]. For the AWARE protocol this entails that at least t of the n custodians must be available and that r of the q auditing services must be available. It is, therefore, not enough to define that t or more custodians and r or more auditing services are compromised. The relevant aspect for availability is whether t or more custodians and r or more auditing services are compliant with the protocol. If so, the availability is guaranteed. If, however, less than t custodians or less than r auditing services are trustworthy the system locks itself. This is a denial of service (DOS) as the PD_u is not accessible anymore. By introducing h, the maximum of restore releases, the policy prohibits further publication of auditing shares which can result in an even earlier DOS. However, both DOS's are expected as they block an illegitimate restore attempt. During such a DOS, the user can be in one of two states: State one, the user is still in possession of his private data. Then there is no problem, he should simply choose more trustworthy custodians after seeing the auditing trail on the DL. State two, the user is not in possession of his private data, and the h published auditing shares contain less than r key shares. Then the protocol has locked access. When this situation occurs, the private data is inaccessible. This behavior is consistent with the requirements, that any entity other than the user has a very low likelihood of accessing the backed up private data while the user has a high likelihood. This high likelihood allows for lost access. Therefore, such an DOS is exactly what we are aiming for. If an illegitimate restore occurs locked access is the preferable state to identity theft and compromise. To prevent this still unpleasant situation users are advised to monitor restore requests on the DL to be aware of such illegitimate restore attempts and be able to engage in time.

Risk. Table 3 shows the numbers of our risk analysis. We conduct it with $t = 3$ of the $n = 5$ custodians being required to restore, as well as $r = 3$ of the $q = 5$ services being required for restore. The amount of legitimate restore responses is limited to $h = r$, and we choose $p = 100$ services in total. Our risk analysis's terminology and scale are based on the NIST 'Risk Management Guide' [28].

A compromise can concern the custodians, the shares stored at the custodians, the service, the low entropy information ρ or a combination of them. We assume a compromise if at least t custodians or shares are compromised. All compromises can either occur or not, except the compromise of the auditing services. Here we distinguish not occurring, the compromise of r services and

the compromise of all services. We do this to account for the risk in a situation in which ρ is compromised. We further distinguish the custodian and the share, because a compromise of the custodians would enable the adversary to send sufficiently many valid promote message on the DL to conduct a restore, while a compromise of the shares would not. After calculating the risk rating for the five isolated compromises we calculate the risk rating of their mutual occurrence.

The likelihood is determined per isolated compromise. The low entropy information ρ of the user can be extracted by physical theft, social engineering or simply by a random guess. The likelihood of physical theft is zero as ρ gets only memorized by the user. The likelihood of social engineering can only be approximated. Even though the low entropy information is a novel concept, it can be compared to a user's password. Happ, Melzer and Steffgen found in [14] that 38.6% of the students they interviewed are willing to give up their password just by talking less than two minutes to female interviewers. In the case that a treat is provided directly before the password is asked, the likelihood rises to 47.8%. Therefore, we decided to model the likelihood of social engineering with 0.4. The likelihood of a random guess is $\frac{q!(p-r))!}{(q-r)!p!}$ as all r service must be chosen correctly from all p services. Therefore, we assume it to be negligible if p is chosen properly. For the numbers used in our risk analysis the likelihood of a random guess is $\frac{5!(100-3))!}{(5-3)!100!} \approx 0.000062$. The share stored at a custodian can only be extracted by social engineering. This is the case as the device is assumed to be offline and the information is stored securely. If the device would be stolen, a full breach must be considered. Therefore, we determine the likelihood of just the share being compromised by 0.4. The share stored at a custodian and the secret signing key of the custodian can, however, be compromised by either social engineering or theft. The likelihood of social engineering is again 0.4. The likelihood of theft is harder to quantify. We set it to conservative 0.5, assuming that the chances are fifty-fifty. Those two events are not mutually exclusive, therefore, the accumulated likelihood of the custodian being fully compromised is 0.7. The compromise of a service is quite unlikely. However, we set it to the same conservative 0.5, again assuming that the chances are fifty-fifty as the data is out of control of the user. All other likelihoods used in the table are combinations of these base scenarios.

The impact can be low, medium or high. A high impact describes a situation in which all knowledge is accessible to the adversary (PD_u can be reconstructed). The medium impact describes a situation in which one piece of information is missing to the adversary. The low impact describes a situation in which more than one piece of information is missing to the adversary.

Seven mutual compromises result in a potential break of the system. Only one of them, however, has a risk rating which exceed *very low* by being *low*. The second *low* risk rating is from a moderate impact. It is crucial that both of these events include compromise of the custodians, thus compromise of the share and the capability to promote a new identity on the DL. The risk rating of just the compromise of the custodians is larger than the risk rating of compromise of the custodians and the low entropy information ρ, due to a higher likelihood.

Compromised				Likelihood	Impact	Risk Rating for	
Custodians	Shares	Service	ρ			t,r = 3 and p = 100	
No	No	No	Yes	0.4 [14]	Low(10)	4	Very low
No	Yes	No	No	0.4^t	Low(10)	0.64	Very low
Yes	No	No	No	0.7^t	Medium(50)	17.14	Low
No	No	r	No	0.5^r	Low(10)	1.25	Very low
No	No	All	No	0.5^p	Medium(50)	$4 * 10^{-29}$	Very low
No	No	r	Yes	$0.4 * 0.5^r$	Medium (50)	2.5	Very low
No	No	All	Yes	$0.4 * 0.5^p$	Medium (50)	$1.58 * 10^{-29}$	Very low
No	Yes	No	Yes	$0.4^t * 0.4$	Medium (50)	1.28	Very low
No	Yes	r	No	$0.4^t * 0.5^r$	Medium(50)	0.4	Very low
No	Yes	r	Yes	$0.4^t * 0.5^r * 0.4$	High (100)	0.32	Very low
No	Yes	All	No	$0.4^t * 0.5^p$	High (100)	$5.05 * 10^{-30}$	Very low
No	Yes	All	Yes	$0.4^t * 0.5^p * 0.4$	High (100)	$2.02 * 10^{-30}$	Very low
Yes	No	No	Yes	$0.7^t * 0.4$	High (100)	13.71	Low
Yes	No	r	No	$0.7^t * 0.5^r$	Medium(50)	2.14	Very low
Yes	No	r	Yes	$0.7^t * 0.5^r * 0.4$	High (100)	1.72	Very low
Yes	No	All	No	$0.7^t * 0.5^p$	High (100)	$2.71 * 10^{-29}$	Very low
Yes	No	All	Yes	$0.7^t * 0.5^p * 0.4$	High (100)	$1.08 * 10^{-29}$	Very low

Table 3. Risk analysis for compromised custodians, shares, services, ρ and their combinations. The likelihood is determined as described in the text. The terminology and method are based on the NIST Risk Management Guide [28], the *risk rating* = likelihood $*$ impact.

Compared to the other proposed solutions with the same likelihood assumptions it can be seen that the worst risk rating of the AWARE protocol is *low* with a risk rating of 13.71. It is close to the best risk rating of social backup and local backup. In Table 4 these risk ratings are presented. The risk of just trusting

Compromised	Likelihood	Impact	Risk Rating for t = 3	
Social backup	0.7^t	High (100)	34.3	Moderate
Local backup	0.5^d	High (100)	12.5	Low

Table 4. Risk rating for social and device backup.

custodians is much higher with a *moderate* risk rating. Only the *low* risk rating of distributing the private data on local devices is slightly lower than our worst risk rating. However, the guarantees of local backup are highly dependent on its user and do not offer self sovereignty. Our AWARE protocol has a comparable low risk rating while enabling the user to be self sovereign by making use of trusted custodians, auditing service and the assumption that the user is capable of memorizing the low entropy information ρ. To guarantee self sovereignty, the user is required to hold some private information. We reduced its size from the arbitrary sized PD_uvia a fixed size key to the size of ρ.

5 Implementation

We implemented a simulation of the AWARE protocol. ***.

*** https://github.com/phil-jakubeit/aware

We chose the high security level of our instantiation of the AWARE protocol to be 256-bit. The building block of symmetric encryption is instantiated as 256-bit AES [8] in Galois counter mode (GCM) [22]. The building block of the asymmetric encryption and decryption is instantiated as an integrated encryption scheme (ECIES) [20] of the symmetric AES256GCM and the P-521 elliptic curve being specified in NIST FIPS-186-4 [24]. The building block of signing and verifying signatures is instantiated as the elliptic curve digital signature algorithm (ECDSA) [18] over the P-521 curve. The building block of secret sharing is instantiated by Shamir's secret sharing [27] with a modulus of $2^{521} - 1$, the 13th Mersenne prime [31]. The random permutation of the auditing service list is realized by the Fisher-Yates-Durstenfeld random permutation [13].

5.1 Experimentation

We assume 1KB of private data PD_u and an auditing service list $\mathcal{AS}_u$ consisting of the addresses and the public signing keys of the $p = 100$ auditing services. The address is 128-bit, the size of an IPv6 address [15]. The public signing key is 526-bit due to compression of the point on the P-521 curve. Thus is the service list about 8KB. Due to formatting overhead the user and each of his custodians must exchange 24KB in our simulation. The NFC specification [16] and serial infrared communication [3] allow a baud rate of 115,200 bit per second (14.4B/s). With this baud rate the backup exchange and the restore exchange between a user and one custodian each takes less than 2 seconds.

The most cost intensive building blocks we use are the ECC computations on the P-521 curve and the generation of random coefficients for the secret sharing.

The most cost intensive operation of ECC is the scalar multiplication with a time complexity of $\mathcal{O}(n^{k+1})$ for $k = 2$ if ordinary school book multiplication is used and $n = 2^{521} - 1$,the order of the finite field. The implementation of the scalar multiplication under the P-521 curve on the low end ARMv6-M architecture by [19] is optimized for memory efficiency and takes about 84 million cycles. On a 48Mhz processor this equals about 1.7 seconds for one scalar multiplication. Even though the channel between a user and his custodian is secure one signature from the custodian and one verification from the user is required for the user to know the custodians capability to sign with his private key. This adds up to about 8 seconds but the required time can be decreased by using speed optimized implementations or a more powerful microcontroller. The secret sharing requires 521-bit randomness for each random coefficient. Due to the structure of Shamir's secret sharing we require $t-1$ 521-bit strings. The built-in random number generator of an ARM cortex M4 with the ARMv7-M architecture is capable of generating a 32-bit random number every 40 cycles [2]. With $t = 3$ from our risk analysis 1042-bit of randomness are required. This can be generated in 1320 cycles, which takes about 0.0000275 seconds if the processor runs at 48Mhz.

The communication with the DL has costs and is kept to a minimum. Each custodian writes a signed promotion message on the DL. It consists of the hex representation of the ASCII string "SSIAWARE-PROMOTION", the public

signing key of the custodian, the public signing key of the user to restore and the public encryption key of the new user. The hex number requires 18-Bytes, each key requires 66-Bytes in its compressed form, and the signature consists of 132-Bytes. This adds up to 348-Byte of raw information. In our simulation we have up to 785-Bytes due to overhead from the serialized formatting.

The key k is 256-bit, but our secret-sharing scheme operates modulo the 13th Mersenne prime. Therefore, each share of the key can be up to 521-bit. The policy consists of two small integers, h and y and contains further x keys each of size 526-bit. Assuming $y = 3$ and $x = 5$, a complete message to an auditing service can be send in less than 3KB and should be secured via TLS [10].

A signed restore release message consists of the hex representation of the ASCII string "SSIAWARE-RESTORE", the public signing key of the user to restore, the public signing key of the auditing service and the encrypted auditing share. The hex number requires 16-Bytes, each key requires 66-Bytes, the encrypted share also requires 66-Bytes, and the signature consists of 132-Bytes. This adds up to 280-Byte of raw information. In our simulation we have up to 1KB due to serialized formatting.

6 Discussion

We achieved a high likelihood for the user to restore and a low likelihood for every other entity by requiring public commits to a DL.

Data can be written on any ledger. Exemplary for the Bitcoin (BC) ledger the data is concatenated, padded and extended with its RIPEMD-160 [11] hash. Then it is split into 20-Byte chunks and the minimum amount of BC is send to each 20-Byte chunk (BC address) [5]. The minimum amount of a single transaction is 0.00000547 BC and the fee per byte is 0.00000014 BC.

For the conservative amount of 785-Bytes in our simulation this equals 0.00032492 BC (3.04 Euro for 9348.66BC/Euro) for each promotion message. Each auditing service as_j that writes a restore release to the DL is must spend 0.0004278BC (3.95 Euro for 9348.66BC/Euro). The amount of custodians required for restore and the amount of auditing services required for restore directly translate to costs. Exemplary Figure 4 shows that the amount of custodians directly translates to costs and to the risk rating of the AWARE protocol. The thresholds show that an increase in the number of custodians reduces the worst risk rating to *very low* with the tradeoff that the costs rise above 20 Euro. It also shows that less than 3 custodians imply a rise of the risk rating to *moderate*.

Other DL like the Ethereum ledger allow for smart contracts. Smart contracts can be used to write the promotion message and the restore release directly on the ledger. They can, however, also interact smart, thus performing gated re-encryption. Auditing services would be required to have a specific key per user. The user sends data to the smart contract which validates the policy and performs re-encryption in case of a valid restore request. This would shift the computation from the auditing services to the smart contracts for the increased price of computation on the DL, while granting compliance to the policy.

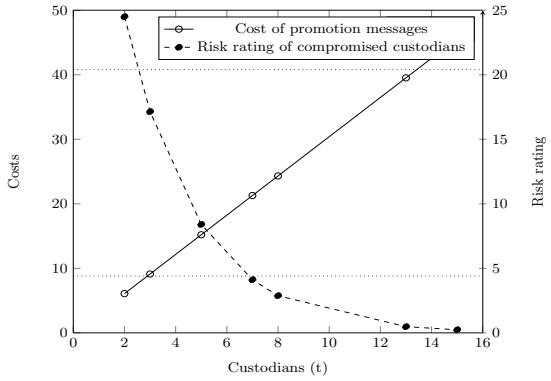

Fig. 4. Plot of the cost of promotion messages with varying t custodians and the worst risk rating of AWARE. The dotted lines indicate the threshold for the risk rating *Very Low* (0-4), *Low* (5-20) or *Moderate* (21-79) [28].

A private ledger (e.g. Hyperledger [29]) of trusted nodes could also be used. This decreases the cost to the power consumption of the auditing services. It must further be guaranteed that the ledger is accessible to any participant without authentication.

7 Conclusion and Future work

Our AWARE protocol is built on physical trust relations towards custodians and handles a potential compromise by requiring online auditing on a DL. The exclusive information ρ in combination with the policy π_u empowers the user to be the only entity capable of restoring his backed up data with certainty. This makes the AWARE protocol the only self-sovereign backup-and-restore protocol. Backup-and-restore mechanisms in the literature make the trust assumption either towards a TA, towards other participants or towards devices. We make use of all these concepts, the auditing services, the custodians and the ISTs. However, in the AWARE setting the user is sovereign. The only trust assumption left is that the auditing services abide to the policy specified by the user. Future research can look into concepts of enforcing the policy (e.g. smart contracts [6] or trusted execution environments [25]). It is further desired to speed optimize ECC computations on low end 32-bit microcontrollers for security parameters of 256-bit.

The private data is held by the custodians and even in case of compromise of multiple custodians there is no possibility to retrieve the private data without notice. The auditing services involved are assumed to abide to the policy, therefore, either the symmetric cipher used (AESGCM256) must be broken or promotions on the DL are required to decrypt the private data. These promotions reveal the identity of the compromised custodians and the user can engage. The user remains in charge by being able to access the auditing log and perform the access authorization. By just holding ρ instead of his private data the user remains self sovereign.

References

1. A. Abraham. Self-sovereign identity. Styria. EGIZ.GV.AT, 2017.
2. E. Alkim, P. Jakubeit, and P. Schwabe. NewHope on ARM cortex-M. SPACE. Springer, 2016.
3. J. Angerstein. Serial infrared specification. `http://berk.tc/intercon/irda/IrPHY_1p4.pdf`.
4. T. S. Bryan Pon, Chris Locke. Private-sector digital identity in emerging markets.
5. CG. Cryptograffiti, 2019. `https://cryptograffiti.info/`.
6. K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. Access, IEEE, 2016.
7. ConsenSys AG. uPort, 2017. `https://www.uport.me/`.
8. J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.
9. J. R. David Reinsel, John Gantz. The digitization of the world - from edge to core, 2018.
10. T. Dierks. TLS v 1.2, 2008. `http://www.hjp.at/doc/rfc/rfc5246.html`.
11. H. Dobbertin, A. Bosselaers, and B. Preneel. Ripemd-160. FSE, Springer, 1996.
12. European Commission. Data protection in the EU, 2016. `https://www.eugdpr.org/`.
13. R. A. Fisher and F. Yates. *Statistical tables for biological, agricultural and medical research*. Nature. Oliver and Boyd Ltd, 1943.
14. S. Heikkinen. Social engineering in the world of emerging communication technologies. WWRF, IEEE, 2006.
15. R. Hinden. IPv6 specification, 2017. `http://www.hjp.at/doc/rfc/rfc2460.html`.
16. International Organization for Standardization. Radio frequency parameters for communications at 13,56 mhz.
17. International Organization for Standardization. ISO 27000, 2018.
18. D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). IJISS, Springer, 2001.
19. Z. Liu, H. Seo, A. Castiglione, K.-K. R. Choo, and H. Kim. Memory-efficient implementation of elliptic curve cryptography for the internet-of-things. TDSC, IEEE, 2018.
20. V. G. Martínez, L. H. Encinas, et al. A comparison of the standardized versions of ECIES. IAS, IEEE, 2010.
21. U. Maurer. Information-theoretically secure secret-key agreement by not authenticated public discussion. Eurocrypt, Springer, 1997.
22. D. McGrew and J. Viega. The galois/counter mode of operation (GCM). Indocrypt, Springer, 2004.
23. Miniwatts Data Group. World internet users and population stats, 2019. `https://www.internetworldstats.com/stats.htm`.
24. NIST. *FIPS 186-4–Digital Signature Standard (DSS)*. NIST, 2013.
25. M. Sabt, M. Achemlal, and A. Bouabdallah. Trusted execution environment: what it is, and what it is not. ISPA, IEEE, 2015.
26. R. S. Sandhu and P. Samarati. Access control: principle and practice. CM, IEEE, 1994.
27. A. Shamir. How to share a secret. Communications, ACM, 1979.
28. G. Stoneburner, A. Goguen, and A. Feringa. *Risk Management Guide*. NIST, 2012.
29. The Linux Foundation. Hyperledger. `https://www.hyperledger.org/`.
30. The Sovrin Foundation. Sovrin, 2017. `https://sovrin.org/`.
31. E. W. Weisstein. Mersenne prime, 2019. `http://mathworld.wolfram.com/MersennePrime.html`.