Agent Authorization Policy

Version 2

Each Identity Owner creates a Policy on the ledger, identified by an address I. Each agent has an agent policy keypair that will be used with I. The policy allows a key to have some combination of four authorizations:

- **0: PROVE**: agents with this authorization can create Proofs from Credentials
- **1: PROVE_GRANT**: agents with this authorization can grant the PROVE authorization to other agents
- **2: PROVE_REVOKE**: agents with this authorization can revoke the PROVE authorization from other agents
- **3: PROVE_ADMIN**: keys with this authorization can modify the provision or revoke prove authorization policy, i.e., grant and revoke the PROVE_GRANT authorization
    - PROVE_GRANT_GRANT
    - PROVE_GRANT_REVOKE

Each authorization is either on or off. Therefore, an array of authorizations can be represented in a bitmap using the well known position for each authorization. For example in big endian, 0001 represents [PROVE], and 1000 would represent [PROVE_ADMIN].

This is a public record, but no information in this public record is ever shared with any other party. Its purpose is to allow for key management of devices in a flexible way, while allowing for entities to prove in zero knowledge that they are using an agent that is authorized by the entity. This ZKP is possible because the ledger maintains a global accumulator for all keys with the PROVE authorization, called the Prover Registry. When a key is added to a Policy, and that key is given the PROVE authorization, the ledger adds a commitment to the Prover Registry. When a key loses its PROVE authorization, the ledger removes the associated commitment from the Prover Registry.

[TODO: How fine grained should we allow policy management to be?]

- A can revoke A, B, and C, and where B, can only revoke B and C
- This is a deep subtopic

Setup

1. Trusted setup
    a. There is a global accumulator needed to store commitments to the keys with PROVE authorizations. From this accumulator, a key can prove in zero knowledge that it is authorized to prove. This requires a trusted setup.

ID Owner's First Agent

1. Agent creates policy management keypair (A_p^{pk}, A_p^{sk}) and a secret S
2. Agent creates policy address ***I*** on the ledger and adds A_p^{pk} as a key and ensures that key has all authorizations (111).
3. Agent adds K = Comm(S, r0) to the PROVE authorization.
    a. What kind of accumulator to use?
4. The ledger adds commitment Comm(K, ***I***) to the global accumulator ***A***. The only commitments added to the global accumulator are those values with the PROVE authorization.

ID Owner's Subsequent Agents

1. One authorized agent (or multiple agents in a multisig scenario) provisions a new agent.
2. The new Agent creates policy management keypair (A_p^{pk}, A_p^{sk}) and S.
3. New agent sends A_p^{pk} and K= Comm(S, r0) back to the provisioning agent(s).
4. Provisioning agent(s) adds authorization for A_p^{pk} and K to the policy.

Lifecycle

As agents are granted PROVE authorization in address ***I*,** the ledger adds or removes the commitments to the accumulator. Agents can be added to be provisioners and revokers by admin agents. The provision or revoke policy can be changed to require more than one agent to agree on a change like 2 of 3. The ledger will enforce these rules by requiring multiple signatures complete the transaction.

Claim Lifecycle

Issuance

1. Claim Receiver sends a Claim Request, which contains a blinded link secret and blinded address ***I***.
2. Issuer selects an index i for the claim from his non-revocation accumulator A_I.
3. Issuer generates claim C using A_I.
4. Issuer sends C and A_I to Claim Receiver.
5. Issuer adds i to his non-revocation accumulator A_I.
6. Claim Receiver provides C and A_I to Identity Owner.
7. Identity Owner gives C, and A_I to Proof Presenter.

Proof Presentation

1. Claim Presenter refreshes the revocation data from the non-revocation accumulator and Prover Registry.
2. Claim Presenter requests access from a Verifier.
3. Verifier sends a Proof Request: what must be proven and which attributes must be

disclosed.

4. Claim Presenter sends disclosed claim attributes and other proofs (in zero-knowledge) and a zero-knowledge proof that
   a. $K \leftarrow \mathrm{Comm}(S, r_0)$
   b. $C_1 \leftarrow \mathrm{Comm}(K, I)$
   c. $C_2 \leftarrow \mathrm{Comm}(S, r_1)$
   d. $C_3 \leftarrow \mathrm{Comm}(K, r_2)$
   e. $C_4 \leftarrow \mathrm{Comm}(C_1, r_4)$
   f. $\mathcal{A}$ is the accumulator that contains C_1
      i. $\mathcal{A}[u, N](b_1, b_2, \ldots, b_n)$ is defined for prime $b_i \in [B; B^2 - 1]$
      ii. $\mathcal{A} \leftarrow u^{b_1 b_2 \cdots b_n} \bmod N$
      iii. We say that $b_i \in \mathcal{A}$ for $b_1, b_2, \ldots, b_n$.
      iv. It must hold that $B^2 - 1 < q/2$
   g. $\mathrm{wit} \leftarrow \mathcal{A}^{-C_1} \bmod N$
   h. Our intention is to prove the hierarchy of commitments: C_3 is a commitment to K, which is itself a commitment to S. We want to prove the knowledge of S by putting it in another commitment C_2.
   i. All values not enclosed in ()'s are assumed to be known to the verifier.
   j. We want to prove
   k. $ZKPoK\{(S, K, C_1, I, r_0) : \quad I \in \mathrm{Claim} \wedge C_1 \in \mathcal{A} \wedge K = \mathrm{Comm}(S, r_0)\}$
   l. For this we need additional commitments C2,C3,C4.

$$\pi \leftarrow \mathrm{DComm}_S(C_2, C_3) \wedge \mathrm{DComm}_K(C_3, C_4) \wedge CommAcc_{C_1}(C_4, \mathcal{A}) \wedge I \in$$

   m. $Claim$
   n. The proof can be prepared as concatenation of

NIZK{ (r4,C1): (C1 in A) && C4 = Comm(C1,r4) }   = CommAcc proof
NIZK{ (K,I,r2): Credential(I) && C3 =Comm(K,I,r2) } = Claim+commitment proof
NIZK{ (K,I,r2,r4): C3 = Comm(K,I,r2) && C4 = Comm(Comm(K,I),r4) }   = DComm proof
NIZK{ (S,r0,r1,r2,I): C2 = Comm(S,r1) && C3 = Comm(Comm(S,r0),I,r2)} = DComm proof

These 4 proofs can be united into a single one with some reduction in size.

   o. Claim index i has not been revoked yet.

5. Claim Presenter generates s', k', i'
6. Claim Presenter computes
   a. $v \leftarrow \mathrm{Comm}(s', k', i')$
   b. $c \leftarrow H(C_2 || C_3 || C_4 || v)$
   c. $\widetilde{s} \leftarrow s' - Sc$
   d. $\widetilde{k} \leftarrow k' - Kc$
   e. $\widetilde{i} \leftarrow i' - Ic$

7. Claim Presenter sends $\{C_2, C_3, C_4, \widetilde{s}, \widetilde{k}, \widetilde{i}, c\}$ to Verifier
8. Verifier validates the proof
    a. Computes
        i. $\overline{v} \leftarrow \mathrm{Comm}(\widetilde{s}, \widetilde{k}, \widetilde{i}) {C_2}^c {C_3}^c {C_4}^c$
        ii. $\overline{c} \leftarrow H(C_2 || C_3 || C_4 || \overline{v})$
    b. Verifies that $\overline{c} = c$

“I am a valid claim presenter”<=
<=“I am part of the claim policy”<=
<=“My public key is part of the claim policy”<=
<=“I know the private key of the public key of the claim policy”<=
<=“I know a private key and a public key which are part of the policy that is part of the claim.”<=
<=”I know S which is private key of a public key which is part of the policy that is mentioned in the claim”<=
<=”I know K and S s.t. K is a commitment to S, and K is part of the policy that is mentioned in the claim”<=
<=”I know K and S s.t. K is a commitment to S, and K is part of policy I, and I is the policy in the claim”
<=”I know K and S s.t. K is a commitment to S, and (K,I) belong to the key-policy accumulator in the claim, and I is the policy in the claim”<=
<=”Here is C_1 which is a commitment to K which is a commitment to S, s.t. (K,I) belong to the key-policy accumulator, and I is the policy in the claim”

Potential attacks

1. Inspecting the ledger reveals all of the agents that are associated with an entity. Hence even if A_p^{pk} is revealed for one agent, agent is correlated to the entity, all agents of that entity are correlated.
    a. The security is provided by the fact that A_p^{pk} cannot be derived from or correlated to the proof.
2. An agency that is malicious or compromised (all storage is encrypted). Kinds of compromises:
    a. Agency router logs are leaked, but the router is still in control of agency.
    b. Adversary is able to get into Agency machines and eavesdrop on all events (server logs, read disk files, etc) but still not manipulate agency machines.
    c. Active attack (partial control): The adversary controls and is able to manipulate some part of the agency.
3. It will become obvious to third parties how secure a given address (person or organization) is. Does this encourage an attacker?
4. Correlation by network inspection (source/destination IP address, inference of packet timing and size)
    a. An agency router can act as a privacy-enhancing proxy.

    i. Reason 1, by going through a proxy, the recipient would see the proxy's IP address, not the sender's IP address. Because many are using the same proxy, this masks the sender somewhat.
    ii. Reason 2, an attacker could correlate based on the timing of messages or the size of messages sent through a router. A Router could hold messages for a random period of time before forwarding, and add random number of bytes of garbage to a message to thwart these types of inference attacks.
    iii. Note: this could motivate an attack on a major agency for the purposes of eavesdropping.
b. With this solution, isn't It now *more* imperative that we employ a mix network? Or did we have the same risk with correlation of IP addresses?

Version 1

TO ADD: agents can be linkable within the same relationship (DID-DID), but not across relationships.

UPDATE: Simplification with scheme

0. Introduction

This protocol supports:

- Unique Identity Owner;
- Multiple Indistinguishable Claim Receivers;
- Multiple Distinguishable Claim Presenters (Agents);
- Unique Updateable Agent Revoker;
- Unique Non-updateable Provisioner.

1. Table of assets and roles

Asset\Role	Identity Owner	Provisioner	Claim Receiver	Claim Presenter	Claim Revoker	Agent Revoker	Verifier	Claim Issuer
Agent key a_{ag}				O				
Agent revocation key a_{ar}						O		

Agent ID V_{ag}		KM		O		KM		
Agent Revoker ID [1] V_{ar}		K				O		
Provisioning key a_{pr}		O						
Provisioning ID V_{pr}		O	K	K				
Agent certificate S_a		KM		O				
Link secret a_{ls}	O		K	K				
Claim revocation key a_{cr}					O			
Claim revocation ID V_{cr}			K	K	K			
Owner defined attributes A_O	O		K	K			K[2]	
Issuer defined attributes A_I			K	K			K[3]	O

Legend

O: owns
K: knows
KM: knows multiple values

2. Construction

- (a_{ag}, V_{ag}) - private-public signature keypair, unique for agent.
- (a_{ar},V_{ar}) - private-public signature keypair;
- (a_{pr},V_{pr}) - private-public signature keypair;
- $S_a = Sig_{a\_pr}(V_{ar},V_{ag})$
- (a_{cr},V_{cr}) - private-public signature keypair;
- Claim attributes signed by Issuer: V_{pr},a_{ls}, identity attributes

[1] Can be a threshold signature public key if we want a multisignature revocation
[2] Selectively disclosed
[3] Selectively disclosed

3. Use cases

3.1. Setup

3.1.1 Identity Owner

1. Identity Owner generates link secret a_{ls} as a random value
2. Identity Owner defines owner-defined attributes A_O;
3. Identity Owner selects an agent accumulator: his own one or a global one.

3.1.2 Provisioner

1. Provisioner generates provisioning key a_{pr};
2. Provisioner computes provisioning ID V_{pr};
3. Provisioner tells V_{pr} to Identity Owner.

3.1.3 Claim Receiver

1. Claim Receiver gets V_{pr} from Identity Owner.

3.1.4 Agent Revoker

1. Agent Revoker generates agent revocation key a_{ar} and agent revocation ID V_{ar}.
2. Agent Revoker submits V_{ar} to Provisioner.

3.1.5 Claim Presenter

1. Claim Presenter generates agent key a_{ag} and agent ID V_{ag}.
2. Claim Presenter submits V_{ag} to Provisioner.
3. Provisioner creates agent certificate S_a for Claim Presenter.
4. Provisioner adds S_a to the agent accumulator and sends S_a to Agent Revoker.
5. Claim Presenter stores S_a.

3.2 Claim Lifecycle

3.2.1 Issuance

8. Claim Receiver gets A_O, a_{ls} from Identity Owner;
9. Claim Receiver contacts the Issuer and submits A_O, a_{ls}, and V_{pr} in the blinded form.
10. Issuer selects an index i for the claim and adds it to A_I.
11. Issuer generates claim C using A_I.
12. Issuer sends C and A_I to Claim Receiver.
13. Issues adds i to his non-revocation accumulator.
14. Claim Receiver provides C and A_I to Identity Owner.
15. Identity Owner gives C, A_O, a_{ls},A_I, and V_{pr} to Claim Presenter.

3.2.2 Presentation

9. Claim Presenter refreshes the revocation data from the non-revocation accumulator and agent accumulator.
10. Claim Presenter contacts Verifier.
11. Verifier provides a presentation policy: what attributes must be disclosed.
12. Claim Presenter presents claim attributes (in zero-knowledge) and a zero-knowledge proof that
    a. He knows V_{pr} contained in the claim;
    b. He knows (V_{ar},V_{ag}),S_a, where S_a is a signature of (V_{ar},V_{ag}) on V_{pr};
    c. He knows private key corresponding to V_{ag};
    d. V_{ag} has not been revoked yet.
    e. Claim index i has not been revoked yet.
13. Verifier checks the proof.

3.2.3 Revocation

1. Issuer selects the index i to revoke.
2. Issuer removes i from the non-revocation accumulator

3.3 Agent Revocation

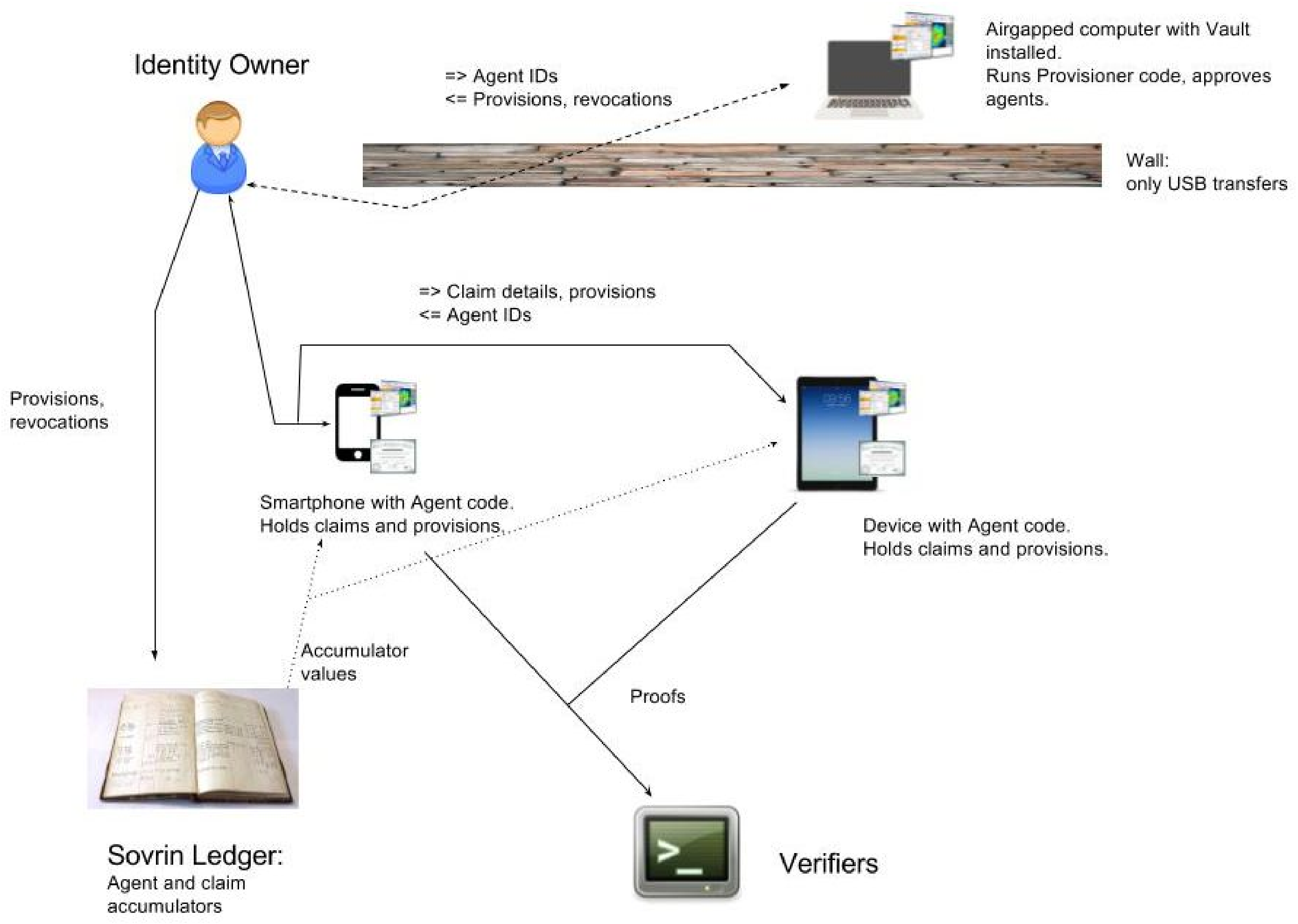
1. Agent Revoker selects the agent he revokes and retrieves his agent ID V_{ag} and certificate S_a.
2. Agent Revoker approaches the agent accumulator and provides S_a and a zero knowledge proof that
    a. S_a is a signature on some key on some (V_{ar},V_{ag}).
    b. He knows the private key from V_{ar}.

3.4 Agent Revoker Rotation

1. Generate new Agent Revoker ID.
2. Issue new agent certificates using the new Agent Revoker ID.
3. Replace the old agent certificates with the old ones.
4. Revoke the old agent certificates/

4. Simplification

The process can be simplified if we assume that the Identity Owner has a software/hardware vault, where he keeps all most valuable secrets and runs protected code. Then the Provisioner code may run there, and needs only occasional interaction with the outer world to publish his ID, provision new agents, and revoke them.


Identity Owner
=> Agent IDs
<= Provisions, revocations
Airgapped computer with Vault installed.
Runs Provisioner code, approves agents.
Wall:
only USB transfers
=> Claim details, provisions
<= Agent IDs
Provisions, revocations
Smartphone with Agent code.
Holds claims and provisions.
Device with Agent code.
Holds claims and provisions.
Accumulator values
Proofs
Sovrin Ledger:
Agent and claim accumulators
Verifiers