Anonymous credentials with revocation

Dmitry Khovratovich

22 September 2016, version 0.15

1 Introduction

1.1 Concept

The concept of *anonymous credentials* allows users of certain web service (for example, online banking) to prove that their identity satisfy certain properties in uncorrelated way without revealing the other identity details. The properties can be raw identity attributes such as the birth date or the address, or a more sophisticated predicates such as “A is older than 20 years old”.

We assume three parties: Issuer, Prover, Verifier (Service). From the functional perspective, the Issuer gives a credential C based on identity X, which asserts certain property $\mathcal{P}$ about X, to the Prover. The identity consists of attributes $m_1, m_2, \ldots, m_l$. The Prover then presents $(\mathcal{P}, C)$ to the Verifier, which can verify that the Issuer had checked that Prover's identity has property $\mathcal{P}$.

For compliance, another party Inspector is often deployed. Inspector is able to deanonymize the Prover given the transcript of his interaction with the Verifier.

1.2 Properties

First of all, credentials are *unforgeable* in the sense that no one can fool the Verifier with a credential not prepared by the Issuer.

We say that credentials are *unlinkable* if it is impossible to correlate the presented credential across multiple presentations. Technically it is implemented by the Prover *proving* with a zero-knowledge proof *that he has a credential* rather than showing the credential.

Unlinkability can be simulated by the issuer generating a sufficient number of ordinary unrelated credentials. Also unlinkability can be turned off to make credentials *one-show* so that second and later presentations are detected.

Credentials are *delegatable* if Prover A can delegate a credential C to Prover B with certain attributes X, so that Verifier would not learn the identity of A if B presents Y to him. The delegation may continue further thus creating a credential chain.

1.3 Pseudonyms

Typically a credential is bound to a certain pseudonym nym. It is supposed that Prover has been registered as nym at the Issuer, and communicated (part of) his identity X to him. After that the Issuer can issue a credential that couples nym and X.

The Prover may have a pseudonym at the Verifier, but not necessarily. If there is no pseudonym then the Verifier provides the service to users who did not register. If the pseudonym nym_V is required, it can be generated from a master secret m_1 together with nym in a way that nym can not be linked to nym_V. However, Prover is supposed to prove that the credential he presents was issued to a pseudonym derived from the same master secret as used to produce nym_V.

2 Simple example

Government (Issuer G) issues credentials with age and photo hash. Company ABC-Co (Issuer A) issues a credential that includes start-date and employment status, e.g., 'FULL-TIME'.

Prover establishes a pseudonym with both issuers. Then he got two credentials independently. After that he proves to Verifier that he has two credentials such that

- The same master secret is used in both credentials;
- The age value in the first credential is over 20;

- The employment status is 'FULL-TIME'.

Steps in Section 4.3 must be executed for each Issuer.

Issuer and Prover mutually trust each other in submitting values of the right format during credential's issuance. This trust can be eliminated at the cost of some extra steps.

3 Functionality

4 Setup

This section describes how the Prover obtains the primary claim and the non-revocation claim from the Issuer.

If there are multiple Issuers, the Prover obtains the claims from them independently. If the Prover wants to present the claims together and link them, he should *chain* the claims by ensuring that every two consecutive primary claims in the chain share at least one attribute of the same value. The simplest way to ensure that is to generate a single master key K and use it as the value of the first attribute m_1 in all primary claims.

4.1 Common parameters for all Issuers

Some parameters are common for all Issuers and are generated as follows.

1. Generate random 256-bit prime ρ and a random 1376-bit number b such that $\Gamma = b\rho + 1$ is prime and ρ does not divide b;

2. Generate random $g' < \Gamma$ such that $g'^b \neq 1 \pmod{\Gamma}$ and compute $g = g'^b \neq 1$.

3. Generate random $r < \rho$ and compute $h = g^r$.

Then (Γ, ρ, g, h) are public parameters.

4.2 Prover setup

Prover generates 256-bit master key K (possibly the same for all Issuers) and set $m_1 = K$.

4.3 Issuer setup

Issuer defines the primary claim schema P with l attributes $m_1, m_2, \ldots, m_l$. The m_1 is reserved for the master secret of the Prover and m_2 is reserved for the context – the enumerator for the provers.

Primary claim setup:

1. Generate random 1024-bit primes p', q' such that $p \leftarrow 2p' + 1$ and $q \leftarrow 2q' + 1$ are primes too. Finally compute $n \leftarrow pq$.

2. Generate a random quadratic residue[1] S modulo n;

3. Select random $x_Z, x_{R_1}, \ldots, x_{R_l} \in [2; p'q' - 1]$ and compute $Z \leftarrow S^{x_Z} \pmod{n}, R_i \leftarrow S^{x_{R_i}} \pmod{n}$ for $1 \leq i \leq l$.

The issuer's public key is $pk_I = (n, S, Z, R_1, R_2, \ldots, R_l, P)$ and the private key is $sk_I = (p, q)$.

Non-revocation claim setup:

1. Fix a pairing function $e(\cdot, \cdot)$ with group G of order q_R.

2. Generate $h, h_0, h_1, h_2, g, \widetilde{h}, u$ as random elements of G and x, sk as random integers $\pmod{q_R}$.

3. Compute

$$pk \leftarrow g^{sk}; \qquad\qquad y \leftarrow h^x.$$

.

4. The issuer revocation public key is $pk_I^R = (q_R, g, h, h_1, h_2, \widetilde{h}, u, pk)$ and the secret key is (x, sk).

The Issuer fixes the number L of credentials per accumulator. For each accumulator:

1. Select γ randomly $\pmod{q_R}$.

[1] using the function *randomQR* from the Charm library.

2. Compute $g_1, g_2, \ldots, g_L, g_{L+2}, \ldots, g_{2L}$ where $g_i = g^{\gamma^i}$.

3. Compute $z = (e(g, g))^{\gamma^{L+1}}$.

4. Set $V \leftarrow \emptyset$, acc $\leftarrow 1$.

The accumulator public key is (z) and secret key is (γ).

5 Issuance of claims

For the new user Issuer selects the accumulator index A_i and the user index i so that (A_i, i) is unique.

0.1 Issuer retrieves the current value acc for accumulator A_i and the set V of issued and non-revoked credential numbers.

0.2 Issuer computes

$$S = A_i||U_i, \quad H_S = H(S) \pmod{2^{256}}.$$

and sets $m_2 = H_S$.

0.3 Issuer sets 256-bit integer attributes $m_3, .., m_l$ for the Prover.

0.4 Issuer generates 80-bit nonce n_0 and sends it to the Prover;

1.1 Prover generates random 2128-bit v' and loads Issuer's key pk_I.

1.1 R Prover loads Issuer's revocation key pk_I^R and generates random $v'_R \pmod{q_R}$.

1.2 Prover computes $U \leftarrow S^{v'} R_1^{m_1} \pmod{n}$ taking S from pk_I.

1.2 R Prover computes $U_R \leftarrow h_2^{v'_R}$ taking h_2 from pk_I^R.

1.4 Prover sends U to the Issuer.

1.4 R Prover sends U_R to the Issuer.

2.1 Issuer generates random 2724-bit number v'' with most significant bit equal 1 and random prime e such that

$$2^{596} \leq e \leq 2^{596} + 2^{119}.$$

2.1 R Issuer generates random numbers $v''_R, c \pmod{q_R}$.

2.2 Issuer computes

$$Q \leftarrow \frac{Z}{US^{v''}(R_2^{m_2} R_3^{m_3} \cdots R_l^{m_l})} \pmod{n}.$$

and

$$A \leftarrow Q^{e^{-1} \pmod{p'q'}} \pmod{n}.$$

2.2 R Issuer computes

$$\sigma \leftarrow \left(h_0 h_1^{m_2} \cdot U_R \cdot g_i \cdot h_2^{v''_R}\right)^{\frac{1}{x+c}}; \qquad w \leftarrow \prod_{j \in V} g_{L+1-j+i}; \tag{1}$$

$$\sigma_i \leftarrow g^{1/(sk+\gamma^i)}; \qquad u_i \leftarrow u^{\gamma^i}; \tag{2}$$

$$\text{acc} \leftarrow \text{acc} \cdot g_{L+1-i}; \qquad V \leftarrow V \cup \{i\}; \tag{3}$$

$$wit_i \leftarrow \{\sigma_i, u_i, g_i, w, V\}. \tag{4}$$

2.3 Issuer sends (A, e, v'') to the Prover.

2.3 R Issuer sends $(i_A, \sigma, c, v''_R, wit_i, g_i, i)$.

2.4 R Issuer publishes updated V, acc.

3.0 Prover computes $v \leftarrow v' + v''$.

3.0 R Prover computes $v_R \leftarrow v'_R + v''_R$.

3.1 R Prover stores *non-revocation claim* $C_2 \leftarrow (i_A, \sigma, c, v_R, wit_i, g_i, i)$.

3.2 R TEST Prover tests

$$\frac{e(g_i, acc_V)}{e(g, w)} \stackrel{?}{=} z; \tag{5}$$

$$e(pk \cdot g_i, \sigma_i) \stackrel{?}{=} e(g, g); \tag{6}$$

$$e(\sigma, y \cdot h^c) \stackrel{?}{=} e(h_0 \cdot h_1^{m_2} h_2^{v_R} g_i, h) \tag{7}$$

Prover stores *primary claim* $C_1 = (\{m_i\}, A, e, v)$.

5.1 Revocation

Issuer revokes ID S, which corresponds to accumulator acc, index i, and valid index set V:

1.1 Issuer sets $V \leftarrow V \setminus \{i\}$;

1.2 Issuer computes $\mathrm{acc} \leftarrow \mathrm{acc}/g_{L+1-i}$.

2.0 Issuer publishes V, acc.

6 Presentation

Prover prepares all claim pairs (C_1, C_2) he wants to submit:

1. Initiates $\mathcal{T}$ and $\mathcal{C}$ as empty sets. Generates random 1024-bit $\widetilde{m_1}$.
2. For all claim pairs (C_1, C_2) executes Section 6.1.
3. Executes Section 6.2 once.
4. For all claim pairs (C_1, C_2) executes Section 6.3.
5. Executes Section 6.3 once.

Then Verifier

1. For all claim pairs (C_1, C_2) executes Section 6.5.
2. Executes Section 6.6 once.

6.1 Initial preparation

Let $\mathcal{A}$ be the set of *all* attribute identifiers present, of which $\mathcal{A}_r$ are the identifiers of attributes that are revealed to the Verifier, and $\mathcal{A}_{\overline{r}}$ are those that are hidden.

Non-revocation proof.

0.0 Verifier sends nonce n_1 to the Prover.

0.1 R Verifier and Prover load Issuer's public revocation key $pk_I^R = (q_R, g, h_1, h, \widetilde{h})$.

0.2 R Prover loads the non-revocation signature (i_A, σ, c)

0.3 R Prover loads g_i, V_{old} from W and sends the number i_A of accumulator to the Verifier.

0.4 R Verifier loads V, acc from Sovrin at index i_A and sends V, acc to the Prover.

0.5 R Prover updates W:

$$w \leftarrow w \cdot \frac{\prod_{j \in V \setminus V_{old}} g_{L+1-j+i}}{\prod_{j \in V_{old} \setminus V} g_{L+1-j+i}};$$
$$V_{old} \leftarrow V.$$

1.0 R Prover selects random $\rho, \rho', r, r', r'', r''', o, o' \pmod{q_R}$;

1.1 R Prover computes

$$E \leftarrow h^{\rho}\widetilde{h}^{o} \qquad D \leftarrow g^{r}\widetilde{h}^{o'}; \tag{8}$$

$$A \leftarrow \sigma\widetilde{h}^{\rho} \qquad \mathcal{G} \leftarrow g_i\widetilde{h}^{r}; \tag{9}$$

$$\mathcal{W} \leftarrow w\widetilde{h}^{r'} \qquad \mathcal{S} \leftarrow \sigma_i\widetilde{h}^{r''} \tag{10}$$

$$\mathcal{U} \leftarrow u_i\widetilde{h}^{r'''} \tag{11}$$

and adds these values to $\mathcal{C}$.

1.3 R Prover computes

$$m \leftarrow \rho \cdot c; \qquad t \leftarrow o \cdot c; \tag{12}$$

$$m' \leftarrow r \cdot r''; \qquad t' \leftarrow o' \cdot r''; \tag{13}$$

2.1 R Prover generates random $\widetilde{\rho}, \widetilde{o}, \widetilde{o'}, \widetilde{c}, \widetilde{m}, \widetilde{m'}, \widetilde{t}, \widetilde{t'}, \widetilde{m_2}, \widetilde{s}, \widetilde{r}, \widetilde{r'}, \widetilde{r''}, \widetilde{r'''}$, $\pmod{q_R}$.

2.2 R Prover computes

$$\overline{T_1} \leftarrow h^{\widetilde{\rho}}\widetilde{h}^{\widetilde{o}} \qquad \overline{T_2} \leftarrow E^{\widetilde{c}}h^{-\widetilde{m}}\widetilde{h}^{-\widetilde{t}} \tag{14}$$

$$\overline{T_3} \leftarrow e(A,h)^{\widetilde{c}} \cdot e(\widetilde{h},h)^{\widetilde{r}} \cdot e(\widetilde{h},y)^{-\widetilde{\rho}} \cdot e(\widetilde{h},h)^{-\widetilde{m}} \cdot e(h_1,h)^{-\widetilde{m_2}} \cdot e(h_2,h)^{-\widetilde{s}} \tag{15}$$

$$\overline{T_4} \leftarrow e(\widetilde{h},\mathrm{acc})^{\widetilde{r}} \cdot e(1/g,\widetilde{h})^{\widetilde{r'}} \qquad \overline{T_5} \leftarrow g^{\widetilde{r}}\widetilde{h}^{\widetilde{o'}} \tag{16}$$

$$\overline{T_6} \leftarrow D^{\widetilde{r''}}g^{-\widetilde{m'}}\widetilde{h}^{-\widetilde{t'}} \qquad \overline{T_7} \leftarrow e(pk \cdot \mathcal{G},\widetilde{h})^{\widetilde{r''}} \cdot e(\widetilde{h},\widetilde{h})^{-\widetilde{m'}} \cdot e(\widetilde{h},\mathcal{S})^{\widetilde{r}} \tag{17}$$

$$\overline{T_8} \leftarrow e(\widetilde{h},u)^{\widetilde{r}} \cdot e(1/g,\widetilde{h})^{\widetilde{r'''}} \tag{18}$$

and add these values to $\mathcal{T}$.

2.3 R TEST Prover tests that for

$$\widetilde{\rho} = \rho \qquad \widetilde{o} = o \qquad \widetilde{o'} = o' \qquad \widetilde{c} = c$$

$$\widetilde{m} = m \qquad \widetilde{m'} = m' \qquad \widetilde{t} = t \qquad \widetilde{t'} = t'$$

$$\widetilde{m_2} = m_2 \qquad \widetilde{s} = v_R \qquad \widetilde{r} = r \qquad \widetilde{r'} = r'$$

$$\widetilde{r''} = r'' \qquad \widetilde{r'''} = r'''$$

the following holds:

$$E \stackrel{?}{=} h^{\widetilde{\rho}}\widetilde{h}^{\widetilde{o}} \qquad 1 \stackrel{?}{=} E^{\widetilde{c}}h^{-\widetilde{m}}\widetilde{h}^{-\widetilde{t}} \tag{19}$$

$$\frac{e(h_0\mathcal{G},h)}{e(A,y)} \stackrel{?}{=} e(A,h)^{\widetilde{c}} \cdot e(\widetilde{h},h)^{\widetilde{r}} \cdot e(\widetilde{h},y)^{-\widetilde{\rho}} \cdot e(\widetilde{h},h)^{-\widetilde{m}} \cdot e(h_1,h)^{-\widetilde{m_2}} \cdot e(h_2,h)^{-\widetilde{s}} \tag{20}$$

$$\frac{e(\mathcal{G},\mathrm{acc})}{e(g,\mathcal{W})z} \stackrel{?}{=} e(\widetilde{h},\mathrm{acc})^{\widetilde{r}} \cdot e(1/g,\widetilde{h})^{\widetilde{r'}} \qquad D \stackrel{?}{=} g^{\widetilde{r}}\widetilde{h}^{\widetilde{o'}} \tag{21}$$

$$1 \stackrel{?}{=} D^{\widetilde{r''}}g^{-\widetilde{m'}}\widetilde{h}^{-\widetilde{t'}} \qquad \frac{e(pk \cdot \mathcal{G},\mathcal{S})}{e(g,g)} \stackrel{?}{=} e(pk \cdot \mathcal{G},\widetilde{h})^{\widetilde{r''}} \cdot e(\widetilde{h},\widetilde{h})^{-\widetilde{m'}} \cdot e(\widetilde{h},\mathcal{S})^{\widetilde{r}} \tag{22}$$

$$\frac{e(\mathcal{G},u)}{e(g,\mathcal{U})} \stackrel{?}{=} e(\widetilde{h},u)^{\widetilde{r}} \cdot e(1/g,\widetilde{h})^{\widetilde{r'''}} \tag{23}$$

Validity proof

0.1 For each non-revealed attribute $i \in \mathcal{A}_{\overline{r}}$ generate random 592-bit number $\widetilde{m_i}$, except for already generated $\widetilde{m_2}$ and $\widetilde{m_1}$

1. For each credential $C = (\mathcal{I} = \{m_j\}, A, e, v)$ and Issuer's public key pk_I:

    1.1 Choose random 2128-bit r;

1.2 Take n, S from pk_I compute

$$A' \leftarrow AS^r \pmod n \text{ and } v' \leftarrow v - e \cdot r \text{ in integers.} \tag{24}$$

Also compute $e' \leftarrow e - 2^{596}$.

2.1 Generate random 456-bit number $\widetilde{e}$ and random 3060-bit number $\widetilde{v}$.

2.2 Compute

$$T \leftarrow (A')^{\widetilde{e}} \left(\prod_{j \in \mathcal{A}_{\overline{r}} \cap \mathcal{I}} (R_i)^{\widetilde{m_j}} \right) (S^{\widetilde{v}}) \pmod n.$$

2.3 Add T to $\mathcal{T}$, A' to $\mathcal{C}$.

2. For each predicate $p:\ m_j \geq z_j$:

(a) Load Z, S from issuer's public key.

(b) Let $\Delta \leftarrow m_j - z_j$ and find (possibly by exhaustive search) u_1, u_2, u_3, u_4 such that

$$\Delta = (u_1)^2 + (u_2)^2 + (u_3)^2 + (u_4)^2;$$

(c) Generate random 2128-bit numbers $r_1, r_2, r_3, r_4, r_\Delta$, compute

$$T_1 \leftarrow Z^{u_1} S^{r_1} \pmod n \qquad T_2 \leftarrow Z^{u_2} S^{r_2} \pmod n \tag{25}$$

$$T_3 \leftarrow Z^{u_3} S^{r_3} \pmod n \qquad T_4 \leftarrow Z^{u_4} S^{r_4} \pmod n \tag{26}$$

$$T_\Delta \leftarrow Z^{\Delta} S^{r_\Delta} \pmod n; \tag{27}$$

and add these values to $\mathcal{C}$.

(d) Take $\widetilde{m_j}$ generated at step 0.1, generate random 592-bit numbers $\widetilde{u_1}, \widetilde{u_2}, \widetilde{u_3}, \widetilde{u_4}$, generate random 672-bit numbers $\widetilde{r_1}, \widetilde{r_2}, \widetilde{r_3}, \widetilde{r_4}, \widetilde{r_\Delta}$ compute

$$\overline{T_1} \leftarrow Z^{\widetilde{u_1}} S^{\widetilde{r_1}} \pmod n \qquad \overline{T_2} \leftarrow Z^{\widetilde{u_2}} S^{\widetilde{r_2}} \pmod n \tag{28}$$

$$\overline{T_3} \leftarrow Z^{\widetilde{u_3}} S^{\widetilde{r_3}} \pmod n \qquad \overline{T_4} \leftarrow Z^{\widetilde{u_4}} S^{\widetilde{r_4}} \pmod n \tag{29}$$

$$\overline{T_\Delta} \leftarrow Z^{\widetilde{m_j}} S^{\widetilde{r_\Delta}} \pmod n; \tag{30}$$

and add this values to $\mathcal{T}$ in the order $\overline{T_1}, \overline{T_2}, \overline{T_3}, \overline{T_4}, \overline{T_\Delta}$.

(e) Generate random 2787-bit number $\widetilde{\alpha}$ and compute

$$Q \leftarrow T_1^{\widetilde{u_1}} T_2^{\widetilde{u_2}} T_3^{\widetilde{u_3}} T_4^{\widetilde{u_4}} S^{\widetilde{\alpha}} \pmod n$$

Add Q to $\mathcal{T}$.

6.2 Hashing

Prover computes

$$c_H \leftarrow H(\mathcal{T}, \mathcal{C}, n_1).$$

and sends c_H to Verifier.

6.3 Final preparation

For primary claim C_1 and non-revocation claim C_2

2.5 R Prover computes

$$\widehat{\rho} \leftarrow \widetilde{\rho} - c_H \rho \pmod{q_R} \qquad \widehat{o} \leftarrow \widetilde{o} - c_H \cdot o \pmod{q_R}$$

$$\widehat{c} \leftarrow \widetilde{c} - c_H \cdot c \pmod{q_R} \qquad \widehat{o'} \leftarrow \widetilde{o'} - c_H \cdot o' \pmod{q_R}$$

$$\widehat{m} \leftarrow \widetilde{m} - c_H m \pmod{q_R} \qquad \widehat{m'} \leftarrow \widetilde{m'} - c_H m' \pmod{q_R}$$

$$\widehat{t} \leftarrow \widetilde{t} - c_H t \pmod{q_R} \qquad \widehat{t'} \leftarrow \widetilde{t'} - c_H t' \pmod{q_R}$$

$$\widehat{m_2} \leftarrow \widetilde{m_2} - c_H m_2 \pmod{q_R} \qquad \widehat{s} \leftarrow \widetilde{s} - c_H v_R \pmod{q_R}$$

$$\widehat{r} \leftarrow \widetilde{r} - c_H r \pmod{q_R} \qquad \widehat{r'} \leftarrow \widetilde{r'} - c_H r' \pmod{q_R}$$

$$\widehat{r''} \leftarrow \widetilde{r''} - c_H r'' \pmod{q_R} \qquad \widehat{r'''} \leftarrow \widetilde{r'''} - c_H r''' \pmod{q_R}.$$

and add them to $\mathcal{X}$.

4.1 Compute

$$\widehat{e} \leftarrow \widetilde{e} + c_H \cdot e';$$
$$\widehat{v} \leftarrow \widetilde{v} + c_H v'.$$

4.2 For all $j \in A_{\overline{r}}$ compute

$$\widehat{m_j} \leftarrow \widetilde{m_j} + c_H m_j.$$

The values $Pr_C = (\widehat{e}, \widehat{v}, \widehat{m_j}, A')$ are the *sub-proof* for claim C_1.

4.3 For each predicate $p: \ m_j \geq z_j$:

(a) For $1 \leq i \leq 4$ compute $\widehat{u_i} \leftarrow \widetilde{u_i} + c_H u_i$.

(b) For $1 \leq i \leq 4$ compute $\widehat{r_i} \leftarrow \widetilde{r_i} + c_H r_i$.

(c) Compute $\widehat{r_\Delta} \leftarrow \widetilde{r_\Delta} + c_H r_\Delta$.

(d) Compute $\widehat{\alpha} \leftarrow \widetilde{\alpha} + c_H(r_\Delta - u_1 r_1 - u_2 r_2 - u_3 r_3 - u_4 r_4)$.

The values $Pr_p = (\{\widehat{u_i}\}, \{\widehat{r_i}\}, \widehat{r_\Delta}, \widehat{\alpha}, \widehat{m_j})$ are the sub-proof for predicate p.

6.4 Sending

Prover sends $(c, \mathcal{X}, \{Pr_C\}, \{Pr_p\}, \mathcal{C})$ to the Verifier.

6.5 Verification

For the claim pair (C_1, C_2) Verifier retrieves relevant variables from $\mathcal{X}, \{Pr_C\}, \{Pr_p\}, \mathcal{C}$.

6.5.1 Non-revocation check

Verifier retrieves the Issuer revocation public key pk_I^R.

1.1 R Verifier computes

$$\widehat{T_1} \leftarrow E^{c_H} \cdot h^{\widehat{\rho}} \cdot \widetilde{h}^{\widehat{o}} \qquad \widehat{T_2} \leftarrow E^{\widehat{c}} \cdot h^{-\widehat{m}} \cdot \widetilde{h}^{-\widehat{t}} \tag{31}$$

$$\widehat{T_3} \leftarrow \left(\frac{e(h_0\mathcal{G}, h)}{e(A, y)}\right)^{c_H} \cdot e(A, h)^{\widehat{c}} \cdot e(\widetilde{h}, h)^{\widehat{r}} \cdot e(\widetilde{h}, y)^{-\widehat{\rho}} \cdot e(\widetilde{h}, h)^{-\widehat{m}} \cdot e(h_1, h)^{-\widehat{m_2}} \cdot e(h_2, h)^{-\widehat{s}} \tag{32}$$

$$\widehat{T_4} \leftarrow \left(\frac{e(\mathcal{G}, \mathrm{acc})}{e(g, \mathcal{W})z}\right)^{c_H} \cdot e(\widetilde{h}, \mathrm{acc})^{\widehat{r}} \cdot e(1/g, \widetilde{h})^{\widehat{r'}} \qquad \widehat{T_5} \leftarrow D^{c_H} \cdot g^{\widehat{r}} \widetilde{h}^{\widehat{o'}} \tag{33}$$

$$\widehat{T_6} \leftarrow D^{\widehat{r''}} \cdot g^{-\widehat{m'}} \widetilde{h}^{-\widehat{t'}} \qquad \widehat{T_7} \leftarrow \left(\frac{e(pk \cdot \mathcal{G}, \mathcal{S})}{e(g, g)}\right)^{c_H} \cdot e(pk \cdot \mathcal{G}, \widetilde{h})^{\widehat{r''}} \cdot e(\widetilde{h}, \widetilde{h})^{-\widehat{m'}} \cdot e(\widetilde{h}, \mathcal{S})^{\widehat{r}} \tag{34}$$

$$\widehat{T_8} \leftarrow \left(\frac{e(\mathcal{G}, u)}{e(g, \mathcal{U})}\right)^{c_H} \cdot e(\widetilde{h}, u)^{\widehat{r}} \cdot e(1/g, \widetilde{h})^{\widehat{r'''}} \tag{35}$$

and adds these values to $\widehat{T}$.

6.5.2 Validity

Verifier uses all Issuer public key pk_I involved into the credential generation and the received $(c, \widehat{e}, \widehat{v}, \{\widehat{m}_i\}, A')$. He also uses revealed m_i for $i \in A_r$. He initiates $\widehat{\mathcal{T}}$ as empty set.

1. For each credential C consider the attributes I used in C and take sub-proof $(\widehat{e}, \widehat{v}, A')$. Then compute

$$\widehat{T} \leftarrow \left(\frac{Z}{\left(\prod_{i \in A_r \cap I}(R_i)^{m_i}\right)(A')^{2^{596}}}\right)^{-c} (A')^{\widehat{e}} \left(\prod_{i \in A_{\overline{r}} \cap I} (R_i)^{\widehat{m_i}}\right) S^{\widehat{v}} \pmod{n}. \tag{36}$$

Add $\widehat{T}$ to $\widehat{\mathcal{T}}$.

2. For each predicate $p: \ m_j \geq z_j$

(a) Using Pr_p and $\mathcal{C}$ compute

$$\widehat{T_i} \leftarrow T_i^{-c} Z^{\widehat{u_i}} S^{\widehat{r_i}} \pmod n \quad \text{for } 1 \le i \le 4; \tag{37}$$

$$\widehat{T_\Delta} \leftarrow (T_\Delta Z^{z_j})^{-c} Z^{\widehat{m_j}} S^{\widehat{r_\Delta}} \pmod n; \tag{38}$$

$$\widehat{Q} \leftarrow (T_\Delta)^{-c} T_1^{\widehat{u_1}} T_2^{\widehat{u_2}} T_3^{\widehat{u_3}} T_4^{\widehat{u_4}} S^{\widehat{\alpha}} \pmod n, \tag{39}$$

and add these values to $\widehat{\mathcal{T}}$ in the order $\widehat{T_1}, \widehat{T_2}, \widehat{T_3}, \widehat{T_4}, \widehat{T_\Delta}$.

6.6 Final hashing

1. Verifier computes
$$\widehat{c_H} \leftarrow H(\widehat{\mathcal{T}}, \mathcal{C}, n_1).$$
2. If $c = \widehat{c}$ output VERIFIED else FAIL.

6.7 Implementation notice

The exponentiation, modulo, and inverse operations are implemented in the Charm library for the class *integer*.

6.8 Why it works

6.8.1 Signature proof

Suppose that the Prover submitted the right values. Then Equation (36) can be viewed as

$$\widehat{T} = Z^{-c} \left(\prod_{i \in A_r} (R_i)^{cm_i} \right) (A')^{\widetilde{e} + c \cdot e' + c2^{596}} \left(\prod_{i \in A_{\overline{r}}} (R_i)^{\widetilde{m_i} + cm_i} \right) S^{\widetilde{v} + cv'}. \tag{40}$$

If we reorder the multiples, we get

$$\widehat{T} = Z^{-c} \left(\prod_{i \in A_r} (R_i)^{cm_i} \right) \left(\prod_{i \in A_{\overline{r}}} (R_i)^{cm_i} \right) (A')^{c \cdot (e' + 2^{596})} S^{cv'} \left(\prod_{i \in A_{\overline{r}}} (R_i)^{\widetilde{m_i}} \right) (A')^{\widetilde{e}} S^{\widetilde{v}} \tag{41}$$

The last three factors multiple to T so we get

$$\widehat{T} = \left(\frac{Z}{(A')^e S^{v'} \prod_i (R_i)^{m_i}} \right)^{-c} T \tag{42}$$

From Equation (24) we obtain that $(A')^e = A^e S^{v - v'}$, so we finally get

$$\widehat{T} = \left(\frac{Z}{A^e S^v \prod_i (R_i)^{m_i}} \right)^{-c} T \tag{43}$$

From the definition of A we get that

$$A^e S^v = \frac{Z}{\prod_i (R_i)^{m_i}}, \tag{44}$$

which implies

$$\widehat{T} = T.$$

6.8.2 Predicate proof

For the proof to be verified, the values $\widehat{T_i}, \widehat{Q}$ derived in Equations (38),(37),(39) must coincide with $\overline{T_i}, Q$ computed by Prover.

We have

$$\widehat{T_\Delta} = (T_\Delta Z^{z_j})^{-c} Z^{\widehat{m_j}} S^{\widehat{r_\Delta}} = Z^{-c\Delta - cz_j + \widehat{m_j}} S^{-cr_\Delta + \widehat{r_\Delta}} = Z^{\widetilde{m_j}} S^{\widetilde{r_\Delta}} = \overline{T_\Delta}. \tag{45}$$

We also have

$$\widehat{T_i} = T_i^{-c} Z^{\widehat{u_i}} S^{\widehat{r_i}} = Z^{-cu_i + \widehat{u_i}} S^{-cr_i + \widehat{r_i}} = Z^{\widetilde{u_i}} S^{\widetilde{r_i}} = \overline{T_i}. \tag{46}$$

Finally,

$$\begin{aligned}
\widehat{Q} &= (T_\Delta)^{-c} T_1^{\widehat{u_1}} T_2^{\widehat{u_2}} T_3^{\widehat{u_3}} T_4^{\widehat{u_4}} S^{\widehat{\alpha}} = \\
&= Z^{-c\Delta} S^{-cr_\Delta} \left(T_1^{\widetilde{u_1}} T_2^{\widetilde{u_2}} T_3^{\widetilde{u_3}} T_4^{\widetilde{u_4}} \right) (T_1^{cu_1} T_2^{cu_2} T_3^{cu_3} T_4^{cu_4}) S^{\widetilde{\alpha} + c(r_\Delta - u_1 r_1 - u_2 r_2 - u_3 r_3 - u_4 r_4)} = \text{Equation (39)} = \\
&= Z^{-c\Delta} S^{-cr_\Delta} Q \, (T_1^{cu_1} T_2^{cu_2} T_3^{cu_3} T_4^{cu_4}) S^{c(r_\Delta - u_1 r_1 - u_2 r_2 - u_3 r_3 - u_4 r_4)} = \\
&= Q Z^{-c\Delta + cu_1^2 + cu_2^2 + cu_3^2 + cu_4^2} S^{-cr_\Delta + r_1 cu_1 + r_2 cu_2 + r_3 cu_3 + r_4 cu_4 + c(r_\Delta - u_1 r_1 - u_2 r_2 - u_3 r_3 - u_4 r_4)} = Q.
\end{aligned} \tag{47}$$