Anonymous Credentials with Range Proofs and Rate Limiting

Jonathan Katz[*]   Mariana Raykova[*]   Samuel Schlesinger[*]

February 14, 2025

1 Overview of this Draft

We provide an overview of our model, including syntactic definitions as well as definitions of security, in Section 2. In Section 3 we describe some preliminaries and building blocks. We give a formal description of our scheme in Section 4.

Note. This draft is intended to give an example of a scheme for illustrative purposes. We have opted for simplicity over efficiency here, and there are several optimizations that can be applied before real-world deployment.

2 Definitions

In our system, we have three types of roles: *issuers*, *clients*, and *verifiers*. We start with functional definitions of a *credential system with range proofs and rate limiting*, followed by definitions of security. Our definitions are for the *privately verifiable* setting where the issuer and verifier are the same entity, but can be extended naturally to the *publicly verifiable* case where verifiers and issuers may be separate.

Definition 1. *A* credential system with range proofs and rate limiting *consists of algorithms/protocols* (KeyGen*,* CredGen*,* ProofGen*,* VrfyProof) *with the following syntax:*

- KeyGen *is run by an issuer to generate keys* $(\mathsf{pk}, \mathsf{sk})$*.*
- CredGen *is an interactive protocol run by an issuer and a client. The issuer has as input its private key* sk *and a (non-negative integer) value* T*, and the client knows the issuer's public key* pk *and* T*. At the end of the protocol, the client outputs a credential* cred*. If* $\mathsf{cred} = \perp$ *it means the client aborted since it detected cheating.*
- ProofGen*, run by a client, takes as input a credential* cred*, a value* T'*, a non-negative epoch number* E*, a bound* B*, and a non-negative index* $i < B$*. It outputs a proof* π*.*
- VrfyProof*, run by the issuer, takes as input a secret key* sk*, a value* T'*, an epoch number* E*, a bound* B*, and a proof* π*; it outputs a bit and a tag* tag*.*

[*]Google.

Correctness requires that if a credential associated with T is used to generate a proof for[1] $T \leq T'$ and $i < B$ then the proof verifies. We also require that tags generated during verification are unique across clients and (E, i) tuples (due to the way rate limiting is done, as discussed below).

Definition 2. *A credential system with range proofs and rate limiting is* correct *if for any efficient adversary $\mathcal{A}$ the probability that* flag *is set to 1 in the following experiment is negligible:*

1. *Generate keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$*, and give* pk *to* $\mathcal{A}$*. Set* $\mathsf{flag} := 0$*.*
2. *$\mathcal{A}$ may interact with the following oracles:*
   - *On the ith query* $\mathsf{CredGen}_{\mathsf{sk}}(T_i)$*, run the credential-generation protocol honestly between the issuer and a client on* T_i*; let* cred_i *be the client's output. If* $\mathsf{cred}_i = \perp$*, set* $\mathsf{flag} := 1$*.*
   - *On query* $\mathsf{ProofGen}(j, T', E, B, i)$*, where* $T_j \leq T'$*,* $i < B$*, and the tuple* (j, E, i) *has not been used in such a query before, run* $\pi \leftarrow \mathsf{ProofGen}(\mathsf{cred}_j, T', E, B, i)$ *and give* π *to* $\mathcal{A}$*. Also run* $(b, \mathsf{tag}) \leftarrow \mathsf{VrfyProof}_{\mathsf{sk}}(T', E, B, \pi)$*. If* $b = 0$ *or* tag *was previously output by* $\mathsf{VrfyProof}$*, set* $\mathsf{flag} := 1$*.*

We consider three security requirements: (1) *soundness* means that if a client (or colluding set of clients) has never received a credential for $T \leq T'$, then it should not be able to generate a convincing proof for T'; (2) *anonymity* means that the issuer should not be able to distinguish two clients who have each received credentials for some value(s) $T \leq T'$; (3) *rate limiting* means that a client should not be able to generate more than B valid proofs for a given epoch without being caught by the issuer. We formalize these below.

Definition 3 (Soundness). *A credential system with range proofs and rate limiting is* sound *if the success probability of any efficient adversary $\mathcal{A}$ in the following experiment is negligible:*

1. *Generate keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$*, and give* pk *to* $\mathcal{A}$*.*
2. *$\mathcal{A}$ is given oracle access to an issuer executing* $\mathsf{CredGen}_{\mathsf{sk}}(\cdot)$*. Let* T^* *denote the minimum value of* T *for which* $\mathcal{A}$ *queried* $\mathsf{CredGen}_{\mathsf{sk}}(T)$*.*
3. *$\mathcal{A}$ outputs* T', E, B, π*, and succeeds if (1)* $\mathsf{VrfyProof}_{\mathsf{sk}}(T', E, B, \pi) = (1, \star)$ *and (2)* $T' < T^*$*.*

Definition 4 (Anonymity). *A credential system with range proofs and rate limiting is* anonymous *if the success probability of any efficient adversary $\mathcal{A}$ in the following experiment is close to* $1/2$*:*

1. *$\mathcal{A}$ outputs* pk*, and then executes two protocols with clients running* $\mathsf{CredGen}(\cdot)$*. Let* T_0, T_1 *be the adversarially chosen inputs to the first and second executions, respectively, and let* $\mathsf{cred}_0, \mathsf{cred}_1$ *be the clients' outputs in the first and second executions.*

   If $\perp \in \{\mathsf{cred}_0, \mathsf{cred}_1\}$*, choose uniform* $b \leftarrow \{0, 1\}$ *and skip to step 5.*
2. *$\mathcal{A}$ can interact with oracles* $\mathsf{ProofGen}(\mathsf{cred}_0, \cdot, \cdot, \cdot, \cdot)$ *and* $\mathsf{ProofGen}(\mathsf{cred}_1, \cdot, \cdot, \cdot, \cdot)$*.*
3. *At some point, $\mathcal{A}$ outputs* T', E, B, i_0, i_1 *with* $T" \geq T_0, T_1$ *and* $i_0, i_1 < B$*, and such that $\mathcal{A}$ never previously queried* $\mathsf{ProofGen}(\mathsf{cred}_0, T', E, \star, i_0)$ *or* $\mathsf{ProofGen}(\mathsf{cred}_1, T', E, \star, i_1)$*. In response, a uniform bit b is chosen, and $\mathcal{A}$ is given* $\mathsf{ProofGen}(\mathsf{cred}_b, T', E, B, i_b)$*.*

[1] The literature typically deals with a client proving that its age T is *at least* some value T'; it is easy to modify our scheme to support that, if desired.

4. *$\mathcal{A}$ can continue to interact with oracles* $\mathsf{ProofGen}(\mathsf{cred}_0, \cdot, \cdot, \cdot, \cdot)$ *and* $\mathsf{ProofGen}(\mathsf{cred}_1, \cdot, \cdot, \cdot, \cdot)$*, but it may not query* $\mathsf{ProofGen}(\mathsf{cred}_0, T', E, \star, i_0)$ *or* $\mathsf{ProofGen}(\mathsf{cred}_1, T', E, \star, i_1)$*.*

5. *At the end of its execution, $\mathcal{A}$ outputs a bit b'. It succeeds if $b' = b$.*

Rate limiting is intended to ensure that each client can generate a bounded number of proofs during an epoch. Ideally, this will be enforced by having the issuer fix a global bound B that is included as part of its public key and used by all clients for all epochs. During each epoch, the issuer will store all the tags it computes as part of proof verification; if, in the course of verifying a proof, the issuer computes a tag it has previously stored (for that epoch), then it knows some client has tried to exceed the bound B (and can reject the proof). Security here thus requires that a client cannot generate more than B proofs with distinct tags in any epoch.

Definition 5 (**Rate limiting).** *A credential system with range proofs and rate limiting is* rate limiting *if the success probability of any* PPT *adversary $\mathcal{A}$ in the following experiment is small:*

1. *Generate keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$*, and give* pk *to $\mathcal{A}$.*

2. *$\mathcal{A}$ is given oracle access to an issuer executing* $\mathsf{CredGen}_{\mathsf{sk}}(\cdot)$*. Let ℓ be the number of times $\mathcal{A}$ queries this oracle.*

3. *$\mathcal{A}$ outputs E, B, and $\{(T'_i, \pi_i)\}_{i=1}^{\ell \cdot B+1}$; let* $(b_i, \mathsf{tag}_i) = \mathsf{VrfyProof}_{\mathsf{sk}}(T'_i, E, B, \pi_i)$*. $\mathcal{A}$ succeeds if (1) $b_i = 1$ for all i and (2) the* $\{\mathsf{tag}_i\}$ *are distinct.*

3 Building Blocks

3.1 (ZK Proofs for) Privately Verifiable BBS Signatures

BBS signatures were introduced implicitly by Boneh, Boyen, and Shacham [3], and were explicitly used for anonymous credentials by Camenisch and Lysyanskaya [5]. Camenisch et al. [4] subsequently showed zero-knowledge (ZK) proofs for partial showings of credentials, based on a variant of the original scheme called BBS$^+$ [1]. Tessaro and Zhu [11] recently showed that the original BBS scheme can be proven secure, and the scheme is now being proposed as a draft standard [9]. We describe a privately verifiable version of the scheme [2, 10] that does not require pairings. We refer to these works for details regarding the computational assumptions needs to prove security.

Fix a group $\mathbb{G}$ of prime order q with generator g, and random elements $h_1, \ldots, h_\ell \leftarrow \mathbb{G}$ for some parameter ℓ. The (privately verifiable) version of BBS signatures then works as follows:

- To generate a secret key, choose $x \leftarrow \mathbb{Z}_q$. The associated public key is $w = g^x$.
- To authenticate a message $(m_1, \ldots, m_\ell) \in \mathbb{Z}_q^\ell$ using secret key x, the issuer chooses $e \leftarrow \mathbb{Z}_q$ and outputs the tag $\left(\left(g \cdot \prod_{i=1}^{\ell} h_i^{m_i}\right)^{1/(e+x)}, e\right)$.
- To verify tag (A, e) on message $(m_1, \ldots, m_\ell) \in \mathbb{Z}_q^\ell$ using secret key x, the issuer checks if

$$A^{e+x} \overset{?}{=} g \cdot \prod_{i=1}^{\ell} h_i^{m_i}.$$

ZK proofs of possession. It is possible to give an efficient ZK[2] proof of possession of a BBS signature on a particular message (without revealing anything else about the signature) to the issuer. In fact, even more is possible. Let $[\ell] = \{1, \ldots, \ell\}$, and $\mathcal{D} \subseteq [\ell]$. Then one can prove to the issuer possession of a signature on a message whose entries in $\mathcal{D}$ are equal to $\{m_i\}_{i \in \mathcal{D}}$, without revealing anything about the entries of the message at indices not in $\mathcal{D}$.

Assume a client has tag (A, e) on message $(m_1, \ldots, m_\ell)$, and let $\mathcal{D}$ be the indices whose entries will be disclosed. Let $B = g \cdot \prod_{i=1}^{\ell} h_i^{m_i}$. The proof works as follows:

1. The client chooses $r_1, r_2 \leftarrow \mathbb{Z}_q^*$ and sets $A' := A^{r_1 r_2}$, $\bar{B} := B^{r_1}$, and $r_3 := r_1^{-1}$.
2. It then sends $\{m_i\}_{i \in \mathcal{D}}$ and $A', \bar{B}$ along with a proof that there exist $\{m_i\}_{i \notin \mathcal{D}}, e, r_2, r_3$ such that[3] (1) $\bar{A} = (A')^{-e} \cdot \bar{B}^{r_2}$ and (2) $H_1 := g \cdot \prod_{i \in \mathcal{D}} h_i^{m_i} = \bar{B}^{r_3} \cdot \prod_{i \notin \mathcal{D}} h_i^{-m_i}$. The proof is done as follows (we describe it as an interactive proof, but it can be made non-interactive using the Fiat-Shamir transform):
   (a) The client chooses $\{m_i'\}_{i \notin \mathcal{D}}, e', r_2', r_3' \leftarrow \mathbb{Z}_q$, and sends to the issuer $A_1 := (A')^{e'} \cdot \bar{B}^{r_2'}$ and $A_2 := \bar{B}^{r_3'} \cdot \prod_{i \notin D} h_i^{m_i'}$.
   (b) The issuer chooses $c \leftarrow \mathbb{Z}_q$ and sends it to the client.
   (c) The client sends $\bar{e} := -c \cdot e + e'$, $\bar{r}_2 := c \cdot r_2 + r_2'$, $\bar{r}_3 := c \cdot r_3 + r_3'$, and $\{\bar{m}_i := -c \cdot m_i + m_i'\}_{i \notin \mathcal{D}}$.
   (d) The issuer computes $\bar{A} = (A')^x$ and $H_1 := g \cdot \prod_{i \in \mathcal{D}} h_i^{m_i}$, and then accepts iff $(A')^{\bar{e}} \bar{B}^{\bar{r}_2} \stackrel{?}{=} \bar{A}^c \cdot A_1$ and $\bar{B}^{\bar{r}_3} \prod_{i \notin \mathcal{D}} h_i^{\bar{m}_i} \stackrel{?}{=} H_1^c \cdot A_2$.

When both parties are honest the issuer always accepts. To see this, note first that

$$(A')^{-e} \bar{B}^{r_2} = (A')^{-e} B^{r_1 r_2} = (A')^{-e} (A')^{e+x} = (A')^x.$$

Furthermore,

$$(A')^{\bar{e}} \cdot \bar{B}^{\bar{r}_2} = (A')^{-c \cdot e + e'} \cdot \bar{B}^{c \cdot r_2 + r_2'} = ((A')^{-e} \bar{B}^{r_2})^c \cdot A_1 = \bar{A}^c A_1$$

and

$$\bar{B}^{\bar{r}_3} \cdot \prod_{i \notin D} h_i^{\bar{m}_i} = \bar{B}^{c \cdot r_3 + r_3'} \cdot \prod_{i \notin D} h_i^{-c m_i + m_i'} = H_1^c \cdot A_2.$$

We sketch why the above is a proof of knowledge of a tag. To show this, we assume the knowledge extractor has access to a DDH oracle that, given a pair (U, V), returns 1 iff $U^x = V$. Note that given such an oracle, the knowledge extractor can tell when a client's proof is correct (by checking that $(A')^{\bar{e}} \bar{B}^{\bar{r}_2} / A_1 = ((A')^c)^x$). This allows the knowledge extractor to compute $\{m_i\}_{i \notin \mathcal{D}}, e, r_2, r_3$ as well as $\bar{A} = (A')^x$ such that $\bar{A} = (A')^{-e} \bar{B}^{r_2}$ and $g \cdot \prod_{i \in [\ell]} h_i^{m_i} = \bar{B}^{r_3}$. If $r_2 = 0$ then $x = -e$ and a valid tag can be computed. If $r_2 \neq 0$ then these equations imply that

$$(A')^{r_3 / r_2} = \left(g \cdot \prod_{i \in [\ell]} h_i^{m_i}\right)^{1/(x+e)},$$

and so $((A')^{r_3/r_2}, e)$ is a valid tag.

[2] The proof can be shown to be ZK under certain assumptions, but we remark that ZK is not necessary for pseudonymity (since the weaker notion of witness indistinguishability suffices).

[3] $\bar{A}$ is not computed by the client. But the issuer can compute $\bar{A}$ and check the corresponding claim.

3.2 The Dodis-Yampolskiy PRF

Although the Dodis-Yampolskiy PRF [8] was originally defined as a VRF in a pairing-based group, we can also view it as a PRF in an arbitrary group. The secret key is $k \in \mathbb{Z}_q$. The evaluation of the PRF on input $i \in \mathbb{Z}_q \setminus \{-k\}$ is $g^{1/(k+i)}$. The outputs of the function are conjectured to be pseudorandom even given g^k.

4 A Credential System

We describe a privately verifiable scheme, and then discuss the changes needed to make it publicly verifiable.

4.1 A Privately Verifiable Scheme

System-wide setup. Fix (random) generators $h_1, \ldots, h_4 \in \mathbb{G}$ to be used by all issuers. (These could be generated in an appropriate way using a hash function.) Also fix positive integers C, L, where L is a statistical parameter related to anonymity and C is a computational parameter related to soundness. They cannot be made arbitrarily large, however, since security also requires $T^*C^2 \cdot (1 + 4 \cdot (L+1)^2) < q$, where T^* is a bound on the largest value of T used in the system. We remark that it is possible to decrease L, without sacrificing anonymity, by having the client do more work when generating a proof.

Key generation. An issuer chooses a secret key $x \leftarrow \mathbb{Z}_q$; the associated public key is $w := g^x$. We also assume that bound $B = 2^\ell$ is either fixed system-wide, or included as part of the public key.

Credential generation. To issue a credential for T to some client, the client and issuer run the following protocol:

1. The client chooses $k \leftarrow \mathbb{Z}_q$, sets $K := h_2^k$, and computes a non-interactive proof of knowledge of k as follows:

   (a) Choose $k' \leftarrow \mathbb{Z}_q$ and set $K_1 := h_2^{k'}$.

   (b) Compute $\gamma := H(K \| K_1)$, and set $\bar{k} := \gamma \cdot k + k'$.

   The client sends $K, \gamma, \bar{k}$ to the issuer.

2. The issuer computes $K_1 := h_2^{\bar{k}} \cdot K^{-\gamma}$ and checks that $H(K \| K_1) \stackrel{?}{=} \gamma$.

3. The issuer then chooses $e \leftarrow \mathbb{Z}_q$ and sends $(A, e) = \left(\left(g \cdot h_1^T \cdot K \right)^{1/(e+x)}, e \right)$ to the client. The issuer also generates a proof that it computed this correctly, by proving that $\log_A \left(g \cdot h_1^T \cdot K \right) = \log_g \left(g^e \cdot w \right)$ using a standard "equality-of-discrete-logarithms" proof. That is, let $X_A = g \cdot h_1^T \cdot K$ and $X_g = g^e \cdot w$. The issuer does:

   (a) Choose $\alpha \leftarrow \mathbb{Z}_q$ and compute $Y_A := A^\alpha$ and $Y_g := g^\alpha$.

   (b) Compute $\gamma := H(A \| e \| Y_A \| Y_g)$.

   (c) Compute $z := \gamma \cdot (x + e) + \alpha$, and send γ, z to the client.

4. The client verifies the proof γ, z by computing $Y'_A := A^z \cdot X_A^{-\gamma}$ and $Y'_g := g^z \cdot X_g^{-\gamma}$ and then checking if $H(X_A \| X_g \| Y'_A \| Y'_g) \overset{?}{=} \gamma$. If so, the client outputs the credential (A, e, k); otherwise, the client outputs $\perp$.

Proof generation. Let T', E be the time period and epoch agreed upon by the client and issuer, and assume the client holds a credential (A, e, k) on $T \in \{0, \ldots, T'\}$. Let $i < B$ have binary representation $i_{\ell-1} \cdots i_0$. The client then does:

- (PoK of credential.)
  1. Choose $r_1, r_2, e', r'_2, r'_3, \Delta', k', s' \leftarrow \mathbb{Z}_q$.
  2. Set $B := g h_1^T h_2^k$, $A' := A^{r_1 r_2}$, $\bar{B} := B^{r_1}$, $r_3 := r_1^{-1}$, and $\Delta := T' - T \geq 0$. Set $\mathsf{to\_send}_1 := A' \| \bar{B}$.
  3. Compute $A_1 := (A')^{e'} \cdot \bar{B}^{r'_2}$ and $A_2 := \bar{B}^{r'_3} h_1^{\Delta'} h_2^{k'} h_3^{s'}$, and set $\mathsf{to\_hash} := A_1 \| A_2$.

- (PRF + commitments and associated proofs.)
  1. Set $Y := h_2^{1/(k+2^\ell E+i)}$, and $\mathsf{to\_send}_1 := \mathsf{to\_send}_1 \| Y$.
  2. Choose $s_0, \ldots, s_{\ell-1} \leftarrow \mathbb{Z}_q$ and set $\mathsf{com}_j := h_3^{s_j} h_2^{i_j}$ (for $j = 0, \ldots, \ell-1$), $s^* := \sum_{j=0}^{\ell-1} 2^j s_j$, and $\mathsf{to\_send}_1 := \mathsf{to\_send}_1 \| \mathsf{com}_0 \| \cdots \| \mathsf{com}_{\ell-1}$.
  3. For $j = 0, \ldots, \ell-1$ do:
     (a) Set $C_{j,0} := \mathsf{com}_j$ and $C_{j,1} := \mathsf{com}_j / h_2$.
     (b) Choose $r_j, \gamma_j, z_j \leftarrow \mathbb{Z}_q$.
     (c) If $i_j = 0$ set $C'_{j,0} := h_3^{r_j}$ and $C'_{j,1} := h_3^{z_j} C_{j,1}^{-\gamma_j}$.
     If $i_j = 1$ set $C'_{j,0} := h_3^{z_j} C_{j,0}^{-\gamma_j}$ and $C'_{j,1} := h_3^{r_j}$.
     (d) Set $\mathsf{to\_hash} := \mathsf{to\_hash} \| C'_{j,0} \| C'_{j,1}$.
  4. Set $Y_1 := Y^{-k'}$ and $\mathsf{to\_hash} := \mathsf{to\_hash} \| Y_1$.

- (Range proof.)
  1. Compute non-negative integers $y_1, \ldots, y_4$ such that $\Delta = \sum_i y_i^2$.
  2. Choose $r_y, \tilde{r}_y \leftarrow \mathbb{Z}_q$ and $\tilde{y}_1, \tilde{y}_2, \tilde{y}_3, \tilde{y}_4 \leftarrow \{0, \ldots, \sqrt{T'} CL\}$.
  3. Compute $C_y := g^{r_y} \cdot \prod_{i=1}^4 h_i^{y_i}$ and set $\mathsf{to\_send}_1 := \mathsf{to\_send}_1 \| C_y$.
  4. Compute $D_y := g^{\tilde{r}_y} \cdot \prod_{i=1}^4 h_i^{\tilde{y}_i}$ and set $\mathsf{to\_hash} := \mathsf{to\_hash} \| D_y$.
  5. Compute $\alpha := \Delta' - 2 \sum_{i=1}^4 y_i \tilde{y}_i$ and $\tilde{\alpha} := -\sum_{i=1}^4 \tilde{y}_i^2$.
  6. Choose $r_* \leftarrow \mathbb{Z}_q$. Compute $C_* := g^{r_*} h_1^{\alpha}$ and set $\mathsf{to\_send}_1 := \mathsf{to\_send}_1 \| C_*$.
  7. Choose $\tilde{r}_* \leftarrow \mathbb{Z}_q$. Compute $D_* := g^{\tilde{r}_*} h_1^{\tilde{\alpha}}$ and set $\mathsf{to\_hash} := \mathsf{to\_hash} \| D_*$.

- Compute $\gamma := H(\mathsf{to\_send}_1 \| \mathsf{to\_hash})$ for H a suitable hash function with range $\{0, \ldots, C\} \subseteq \mathbb{Z}_q$.

- (Complete PoK of credential.) Compute $z_e := -\gamma e + e'$, $z_{r_2} := \gamma \cdot r_2 + r'_2$, $z_{r_3} := \gamma \cdot r_3 + r'_3$, $z_\Delta := \gamma \cdot \Delta + \Delta'$, $z_k := -\gamma \cdot (k+i) + k'$, and $z_s := -\gamma s^* + s'$. Set $\mathsf{to\_send}_2 := z_e \| z_{r_2} \| z_{r_3} \| z_\Delta \| z_k \| z_s$.

- (Complete proofs for commitments.) For $j = 0, \ldots, \ell-1$ do:

1. If $i_j = 0$ set $\gamma_{j,0} := \gamma - \gamma_j$, $z_{j,0} := \gamma_{j,0} \cdot s_j + r_j$, and $z_{j,1} := z_j$.
   If $i_j = 1$ set $\gamma_{j,0} := \gamma_j$, $z_{j,0} := z_j$, and $z_{j,1} := (\gamma - \gamma_{j,0}) \cdot s_j + r_j$.
2. Set $\mathsf{to\_send}_2 := \mathsf{to\_send}_2 \| \gamma_{j,0} \| z_{j,0} \| z_{j,1}$.

- (Complete range proofs.)

  1. Compute $t_y := \gamma r_y + \tilde{r}_y$ and $z_{i,y} := \gamma y_i + \tilde{y}_i$ for $i = 1, \ldots, 4$.
     Set $\mathsf{to\_send}_2 := \mathsf{to\_send}_2 \| t_y \| z_{1,y} \| z_{2,y} \| z_{3,y} \| z_{4,y}$.
  2. Compute $t_* := \gamma r_* + \tilde{r}_*$. Set $\mathsf{to\_send}_2 := \mathsf{to\_send}_2 \| t_*$.

- The final proof is $\mathsf{to\_send}_1, \gamma, \mathsf{to\_send}_2$.

Verification. The issuer holds T', E, and a secret key x, and receives a proof $\mathsf{to\_send}_1, \gamma, \mathsf{to\_send}_2$. It begins by parsing $\mathsf{to\_send}_1$ as $A' \| \bar{B} \| Y \| \mathsf{com}_0 \| \cdots \| \mathsf{com}_{\ell-1} \| C_y \| C_*$ and $\mathsf{to\_send}_2$ as

$$z_e \| z_{r_2} \| z_{r_3} \| z_\Delta \| z_k \| z_s \| \gamma_{0,0} \| z_{0,0} \| z_{0,1} \| \cdots \| \gamma_{\ell-1,0} \| z_{\ell-1,0} \| z_{\ell-1,1} \| t_y \| z_{1,y} \| z_{2,y} \| z_{3,y} \| z_{4,y} \| t_*.$$

The issuer checks that $A' \neq 1$ and $z_{i,y} \in \{0, \ldots, \sqrt{T'} \cdot C(L+1)\}$ for $i = 1, \ldots 4$, rejecting if these do not hold. Otherwise, it does:

- (Verify PoK of credential.) It sets $\bar{A} := (A')^x$ and $H_1 := g \cdot h_1^{T'} \cdot \left(\prod_{j=0}^{\ell-1} \mathsf{com}_j^{2^j} \right)^{-1}$, and then computes
$$A_1 := (A')^{z_e} \bar{B}^{z_{r_2}} \bar{A}^{-\gamma} \quad \text{and} \quad A_2 := \bar{B}^{z_{r_3}} h_1^{z_\Delta} h_2^{z_k} h_3^{z_s} \cdot H_1^{-\gamma}.$$
  It then sets $\mathsf{to\_hash} := A_1 \| A_2$.

- (Verify proofs of commitments.) For $j = 0, \ldots, \ell - 1$:

  1. Set $\gamma_{j,1} := \gamma - \gamma_{j,0}$, $C_{j,0} := \mathsf{com}_j$, and $C_{j,1} := \mathsf{com}_j / h_2$.
  2. Set $C'_{j,0} := h_3^{z_{j,0}} C_{j,0}^{-\gamma_{j,0}}$ and $C'_{j,1} := h_3^{z_{j,1}} C_{j,1}^{-\gamma_{j,1}}$.
  3. Set $\mathsf{to\_hash} := \mathsf{to\_hash} \| C'_{j,0} \| C'_{j,1}$.

- (Verify PRF proof.) Set $Y_1 := Y^{-z_k} \cdot (h_2 / Y^{2^\ell E})^{-\gamma}$ and $\mathsf{to\_hash} := \mathsf{to\_hash} \| Y_1$.

- (Verify range proof.) Compute $D_y := C_y^{-\gamma} \cdot g^{t_y} \prod_{i=1}^4 h_i^{z_{i,y}}$ and set $\mathsf{to\_hash} := \mathsf{to\_hash} \| D_y$.
  Compute $f_* := \gamma \cdot z_\Delta - \sum_{i=1}^4 z_{i,y}^2$ and $D_* := C_*^{-\gamma} \cdot g^{t_*} h_1^{f_*}$, and set $\mathsf{to\_hash} := \mathsf{to\_hash} \| D_*$.

- Finally, it accepts iff $H(\mathsf{to\_send}_1 \| \mathsf{to\_hash}) \overset{?}{=} \gamma$.

4.2 Adding Public Verifiability

It is simple to make the scheme publicly verifiable using a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$:

1. All group elements used in the previous proof will now be in $\mathbb{G}_1$, with the only exception being that the public key is $w = g_2^x \in \mathbb{G}_2$ for g_2 a generator of $\mathbb{G}_2$.

2. During credential generation, the proof of correctness by the issuer (proving that the credential is valid) is no longer needed. Instead, the client can verify correctness of the credential on its own by checking that $e(A, w) = e\left(A^{-e} g h_1^T K, g_2\right)$.

3. As part of proof generation, the client computes $\bar{A} := (A')^{-e}\bar{B}^{r_2}$ and includes it as part of $\mathsf{to\_send}_1$.

4. As part of proof verification, instead of computing $\bar{A}$ (which is not possible without knowledge of x), the verifier uses the $\bar{A}$ included in $\mathsf{to\_send}_1$. However, it first verifies that value by checking that $e(A', w) = e(\bar{A}, g_2)$.

References

[1] M.H. Au, W. Susilo, and Y. Mu. Constant-size dynamic k-TAA. SCN 2006.

[2] A. Barki, S. Brunet, N. Desmoulins, and J. Traoré. Improved algebraic MACs and practical keyed-verification anonymous credentials. SAC 2016.

[3] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. Crypto 2004.

[4] J. Camenisch, M. Drijver, and A. Lehmann. Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited. TRUST 2016. Available at `https://ia.cr/2016/663`.

[5] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. Crypto 2004.

[6] G. Couteau, D. Goudarzi, M. Klooß, and M. Reichle. Sharp: Short Relaxed Range Proofs. ACM CCS 2022. Available at `https://ia.cr/2022/1153`.

[7] G. Couteau, M. Klooß, H. Lin, and M. Reichle. Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments. Eurocrypt 2021. Available at `https://ia.cr/2021/540`.

[8] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. PKC 2005.

[9] T. Looker, V. Kalos, A. Whitehead, and M. Lodder. The BBS Signature Scheme. Internet Draft draft-irtf-cfrg-bbs-signatures-07, Internet Engineering Task Force, September, 2024. Available at `https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/07`. See also `https://github.com/decentralized-identity/bbs-signature`.

[10] M. Orrù. Revisiting Keyed-Verification Anonymous Credentials. Available at `https://eprint.iacr.org/2024/1552`.

[11] S. Tessaro and C. Zhu. Revisiting BBS Signatures. Eurocrypt 2023. Available at `https://ia.cr/2023/275`.