Anonymous Credentials with Range Proofs and Rate Limiting

Jonathan Katz*, Samuel Schlesinger

February 6, 2025

Abstract

We describe a scheme for privacy-preserving age verification with rate limiting.

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, and proof sketches in Section 5.

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.

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 (i.e., 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 $T' \geq T$ and $i < B$ then the proof verifies. We also require that tags generated during verification are unique across (cred, E, i) tuples (due to the way rate limiting is done).

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 and 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* 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* 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 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 for age verification with rate limiting is* anonymous *if the success probability of any* PPT *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 halt.*
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 $T_0, T_1 \leq T'$ 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* $\mathsf{ProofGen}(\mathsf{cred}_b, T', E, B, i_b)$.

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' and succeeds if $b' = b$.*

Rate limiting is intended to ensure that each client can generate a bounded number of proofs during an epoch. 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. Then, 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 that 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 for age verification with 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 in the work of 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 signature scheme called BBS$^+$ [1]. Tessaro and Zhu [11] recently showed that the original BBS signature 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.

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 integer 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 private key x, the issuer checks if

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

ZK proofs of possession. It is possible to give an efficient ZK[1] 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[2] (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 rejects if $A' = 1$. Otherwise, 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 credential. 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 credential 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 credential.

Alternate proof. Tessaro and Zhu [11, Appendix A] show an alternate proof that is slightly more efficient. Assume again that a client has a credential (A, e) on message $(m_1, \ldots, m_\ell)$, with $\mathcal{D}$ being the indices whose entries will be disclosed. Let $B := g \cdot \prod_{i=1}^{\ell} h_i^{m_i}$. The proof now works as follows:

[1] We do not prove ZK, but will instead prove anonymity of the eventual credential system based on these signatures.

[2] Note that $\bar{A}$ is not computed by the client. But since $\bar{A} = (A')^x$, the issuer can compute $\bar{A}$ and check the corresponding claim.

1. The client chooses $r \leftarrow \mathbb{Z}_q$ and sets $A' := A^r$ and $\bar{B} := (B \cdot A^{-e})^r$. It also computes $r_1 := r^{-1}$ and $e_1 := e \cdot r_1$.

2. It then sends $\{m_i\}_{i \in \mathcal{D}}$ and $A', \bar{B}$ along with a proof that there exist $\{m_i\}_{i \notin \mathcal{D}}, r_1, e_1$ such that $g \cdot \prod_{i \in \mathcal{D}} h_i^{m_i} = (A')^{e_1} \bar{B}^{r_1} \prod_{i \notin \mathcal{D}} h_i^{-m_i}$. The proof is done as follows:

   (a) The client chooses $\{m'_i\}_{i \notin \mathcal{D}}, r'_1, e' \leftarrow \mathbb{Z}_q$ and sends to the issuer $U := (A')^{e'_1} \bar{B}^{r'_1} \prod_{i \notin \mathcal{D}} h_i^{m'_i}$.

   (b) The issuer chooses $c \leftarrow \mathbb{Z}_q$ and sends it to the client.

   (c) The client sends $\bar{e}_1 := c \cdot e_1 + e'_1$, $\bar{r}_1 := c \cdot r_1 + r'_1$, and $\{\bar{m}_i = -c \cdot m_i + m'_i\}_{i \notin \mathcal{D}}$.

   (d) The issuer accepts iff $(A')^x = \bar{B}$ and $U \cdot \left(g \prod_{i \in \mathcal{D}} h_i^{m_i}\right)^c = (A')^{\bar{e}_1} \bar{B}^{\bar{r}_1} \prod_{i \notin \mathcal{D}} h_i^{\bar{m}_i}$.

We first verify correctness. Note first that

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

Moreover,

$$\begin{aligned}
(A')^{\bar{e}_1} \bar{B}^{\bar{r}_1} \prod_{i \notin \mathcal{D}} h_i^{\bar{m}_i} &= (A')^{e'_1} \bar{B}^{r'_1} \prod_{i \notin \mathcal{D}} h_i^{m'_i} \cdot \left((A')^{e_1} \bar{B}^{r_1} \prod_{i \notin \mathcal{D}} h_i^{-m_i}\right)^c \\
&= U \cdot \left((A')^{e/r} \bar{B}^{1/r} \prod_{i \notin \mathcal{D}} h_i^{-m_i}\right)^c \\
&= U \cdot \left(B \cdot \prod_{i \notin \mathcal{D}} h_i^{-m_i}\right)^c = U \cdot \left(g \cdot \prod_{i \in \mathcal{D}} h_i^{m_i}\right)^c.
\end{aligned}$$

We sketch why this is a proof of knowledge of a credential. (As before, we assume the knowledge extractor can verify correct proofs using a DDH oracle.) From the underlying proof, we learn $A', \bar{B} = (A')^x$ and $\{m_i\}_{i \notin \mathcal{D}}, r_1, e_1$ such that $g \cdot \prod_{i=1}^{\ell} h_i^{m_i} = (A')^{e_1} \bar{B}^{r_1} = (A')^{e_1 + r_1 x}$. If $r_1 \neq 0$ then this implies

$$(A')^{r_1} = \left(g \cdot \prod_{i=1}^{\ell} h_i^{m_i}\right)^{1/(x + e_1/r_1)},$$

so $((A')^{r_1}, e_1/r_1)$ is a valid credential for $\{m_i\}$. If $r_1 = 0$ and $e_1 = 0$ then we have $\{m_i\}$ for which $g \cdot \prod_{i=1}^{\ell} h_i^{m_i} = 1$, something which occurs with negligible probability if the discrete-logarithm problem is hard. The remaining case is when $r_1 = 0, e_1 \neq 0$. In that case we get $\bar{B}^{e_1} = \left(g \cdot \prod_{i=1}^{\ell} h_i^{m_i}\right)^x$, something that should also be unlikely (though this needs to be proven).

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 public key (used to prove correct evaluation) is g^k. 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 the public key.

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 $h_1, \ldots, h_4 \in \mathbb{G}$ that will be used by all issuers. (These could be generated in an appropriate way using a hash function.) Also fix positive integers C, L (to be determined later). **Jon's note:** *TBD: give security guidance for C, L.*

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 $H(X_A \| X_g \| Y'_A \| Y'_g) \stackrel{?}{=} \gamma$. 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 $r_1, r_2, e', r'_2, r'_3, \Delta', k', s' \leftarrow \mathbb{Z}_q$. **Jon's note:** *It may be possible to remove s' and related values*
  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 $\mathsf{to\_send}_1 := \mathsf{to\_send}_1 \| \mathsf{com}_0 \| \cdots \| \mathsf{com}_{\ell-1}$. **Jon's note:** *It might be possible to compress these all into one commitment (using more generators), and possibly also bulletproofs. Need to evaluate/discuss*
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 $\mathsf{to\_hash} := \mathsf{to\_hash} \| C'_{j,0} \| C'_{j,1}$.

   Jon's note: *it may be possible to compress these proofs (idea: prove AND via one random linear combination) ...*
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 $H(\mathsf{to\_send}_1\|\mathsf{to\_hash}) \stackrel{?}{=} \gamma$.

Correctness. We now (exhaustively and exhaustingly) verify correctness by showing that corresponding elements in $\mathsf{to\_hash}$ computed by the prover and verifier are equal.

- As in Section 3.1, the value $\bar{A} = (A')^x$ computed by the issuer is equal to $(A')^{-e}\bar{B}^{r_2}$. The client computes $A_1 = (A')^{e'} \cdot \bar{B}^{r'_2}$. The issuer computes
$$\begin{aligned} A_1 = (A')^{z_e}\bar{B}^{z_{r_2}}\bar{A}^{-\gamma} &= \left((A')^{-e}\bar{B}^{r_2}\right)^\gamma (A')^{e'}\bar{B}^{r'_2}\bar{A}^{-\gamma} \\ &= (A')^{e'} \cdot \bar{B}^{r'_2}. \end{aligned}$$
The client computes $A_2 = \bar{B}^{r'_3} h_1^{\Delta'} h_2^{k'} h_3^{s'}$. The issuer computes
$$\begin{aligned} A_2 &= \bar{B}^{z_{r_3}} h_1^{z_\Delta} h_2^{z_k} h_3^{z_s} \cdot \left(g^{-\gamma} h_1^{-\gamma T'} (h_3^{s^*} h_2^i)^\gamma\right) \\ &= \left(\bar{B}^{r_3} h_1^\Delta h_2^{-(k+i)} h_3^{-s^*}\right)^\gamma \bar{B}^{r'_3} h_1^{\Delta'} h_2^{k'} h_3^{s'} g^{-\gamma} h_1^{-\gamma T'} (h_3^{s^*} h_2^i)^\gamma \\ &= \left(B g^{-1} h_1^{-T} h_2^{-k}\right)^\gamma \bar{B}^{r'_3} h_1^{\Delta'} h_2^{k'} h_3^{s'} \;=\; \bar{B}^{r'_3} h_1^{\Delta'} h_2^{k'} h_3^{s'}. \end{aligned}$$
- For $j = 0, \ldots, \ell-1$:

– If $i_j = 0$, the client computes $C'_{j,0} = h_3^{r_j}$ and $C'_{j,1} = h_3^{z_j} C_{j,1}^{-\gamma_j}$, and sets $\gamma_{j,0} = \gamma - \gamma_j$, $z_{j,0} := \gamma_{j,0} \cdot s_j + r_j$, and $z_{j,1} := z_j$. The issuer computes

$$C'_{j,0} = h_3^{z_{j,0}} C_{j,0}^{-\gamma_{j,0}} = h_3^{\gamma_{j,0} s_j + r_j} h_3^{-\gamma_{j,0} s_j} = h_3^{r_j}$$

and $C'_{j,1} = h_3^{z_{j,1}} C_{j,1}^{-\gamma_{j,1}} = h_3^{z_j} C_{j,1}^{-\gamma_j}$.

– If $i_j = 1$, the client computes $C'_{j,0} = h_3^{z_j} C_{j,0}^{-\gamma_j}$ and $C'_{j,1} = h_3^{r_j}$, and sets $\gamma_{j,0} := \gamma_j$, $\gamma_{j,1} := \gamma - \gamma_{j,0}$, $z_{j,0} := z_j$, and $z_{j,1} := \gamma_{j,1} \cdot s_j + r_j$. The issuer computes $C'_{j,0} = h_3^{z_{j,0}} C_{j,0}^{-\gamma_{j,0}} = h_3^{z_j} C_{j,0}^{-\gamma_j}$ and

$$C'_{j,1} = h_3^{z_{j,1}} C_{j,1}^{-\gamma_{j,1}} = h_3^{\gamma_{j,1} s_j + r_j} h_3^{-\gamma s_j + \gamma_j s_j} = h_3^{r_j}.$$

- The client computes $Y_1 = Y^{-k'}$. The issuer computes

$$\begin{aligned} Y_1 = Y^{-z_k} \cdot (h_2 / Y^{2^\ell E})^{-\gamma} &= Y^{\gamma \cdot (k+i)} Y^{-k'} (h_2 / Y^{2^\ell E})^{-\gamma} \\ &= \left(Y^{k + 2^\ell E + i}\right)^{\gamma} Y^{-k'} h_2^{-\gamma} = Y^{-k'}. \end{aligned}$$

- It is easy to verify that $z_{i,y} \in \{0, \ldots, \sqrt{T'} \cdot C(L+1)\}$ for all i. Indeed, $z_{i,y} = \gamma y_i + \tilde{y}_i$ and (1) $\gamma \in \{0, \ldots, C\}$, (2) $y_i \in \{0, \ldots, \sqrt{\Delta}\} \subseteq \{0, \ldots, \sqrt{T'}\}$, and (3) $\tilde{y}_i \in \{0, \ldots, \sqrt{T'} \cdot CL\}$.
  The client computes $D_y = g^{\tilde{r}_y} \cdot \prod_{i=1}^4 h_i^{\tilde{y}_i}$. The issuer computes

$$\begin{aligned} D_y = C_y^{-\gamma} g^{t_y} \prod_{i=1}^4 h_i^{z_{i,y}} &= \left(g^{r_y} \cdot \prod h_i^{y_i}\right)^{-\gamma} g^{\gamma r_y} g^{\tilde{r}_y} \prod h_i^{\gamma y_i} h_i^{\tilde{y}_i} \\ &= g^{\tilde{r}_y} \prod h_i^{\tilde{y}_i}. \end{aligned}$$

- The client computes $D_* = g^{\tilde{r}_*} h_1^{\tilde{\alpha}} = g^{\tilde{r}_*} h_1^{-\sum \tilde{y}_i^2}$. The issuer computes

$$\begin{aligned} D_* = C_*^{-\gamma} g^{t_*} h_1^{f_*} &= g^{-\gamma r_*} h_1^{-\gamma\alpha} g^{\gamma r_* + \tilde{r}_*} h_1^{\gamma z_\Delta - \sum z_{i,y}^2} \\ &= h_1^{-\gamma(\Delta' - 2\sum y_i \tilde{y}_i)} g^{\tilde{r}_*} h_1^{\gamma(\gamma\Delta + \Delta') - \sum(\gamma y_i + \tilde{y}_i)^2}. \end{aligned}$$

  Focusing on the exponent of h_1 in the above, note that

$$\begin{aligned} &-\gamma(\Delta' - 2\sum y_i \tilde{y}_i) + \gamma^2 \Delta + \gamma \Delta' - \sum(\gamma^2 y_i^2 + 2\gamma y_i \tilde{y}_i + \tilde{y}_i^2) \\ &= \gamma^2 \left(\Delta - \sum y_i^2\right) - \sum \tilde{y}_i^2 = -\sum \tilde{y}_i^2. \end{aligned}$$

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 ZK 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 $e(A', w) = e(\bar{A}, g_2)$.

5 (Sketch of) Proofs of Security

For now, we just verify that the scheme in the previous section is a proof of knowledge (of a particular witness we discuss below) and "zero knowledge" (actually, we prove witness indistinguishability). The first suffices to prove soundness and rate limiting, and the second suffices for anonymity.

5.1 Proof of Knowledge Property

We argue that the client-generated proof is *3-special sound*; that is, we show that given three accepting proofs with the same $\mathsf{to\_send}_1, \mathsf{to\_hash}$, and different challenges, we can extract the requisite information. So, assume we have T', E, and

$$\mathsf{to\_send}_1 = A' \| \bar{B} \| Y \| \mathsf{com}_0 \| \cdots \| \mathsf{com}_{\ell-1} \| C_y \| C_*$$

as well as

$$\mathsf{to\_hash} = A_1 \| A_2 \| C'_{0,0} \| C'_{0,1} \| \cdots \| C'_{\ell-1,0} \| C'_{\ell-1,1} \| Y_1 \| D_y \| D_*,$$

and distinct challenges $\gamma, \bar{\gamma}, \hat{\gamma}$ (where without loss of generality $\bar{\gamma}, \hat{\gamma} > \gamma$) with corresponding values

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

$$\bar{z}_e \| \bar{z}_{r_2} \| \bar{z}_{r_3} \| \bar{z}_\Delta \| \bar{z}_k \| \bar{z}_s \| \bar{\gamma}_{0,0} \| \bar{z}_{0,0} \| \bar{z}_{0,1} \| \cdots \| \bar{\gamma}_{\ell-1,0} \| \bar{z}_{\ell-1,0} \| \bar{z}_{\ell-1,1} \| \bar{t}_y \| \bar{z}_{1,y} \| \bar{z}_{2,y} \| \bar{z}_{3,y} \| \bar{z}_{4,y} \| \bar{t}_*,$$

and

$$\hat{z}_e \| \hat{z}_{r_2} \| \hat{z}_{r_3} \| \hat{z}_\Delta \| \hat{z}_k \| \hat{z}_s \| \hat{\gamma}_{0,0} \| \hat{z}_{0,0} \| \hat{z}_{0,1} \| \cdots \| \hat{\gamma}_{\ell-1,0} \| \hat{z}_{\ell-1,0} \| \hat{z}_{\ell-1,1} \| \hat{t}_y \| \hat{z}_{1,y} \| \hat{z}_{2,y} \| \hat{z}_{3,y} \| \hat{z}_{4,y} \| \hat{t}_*$$

for which the proofs verify. Let

$$f_* \stackrel{\text{def}}{=} \gamma z_\Delta - \sum z_{i,y}^2, \quad \bar{f}_* \stackrel{\text{def}}{=} \bar{\gamma} \bar{z}_\Delta - \sum \bar{z}_{i,y}^2, \text{ and } \hat{f}_* \stackrel{\text{def}}{=} \hat{\gamma} \hat{z}_\Delta - \sum \hat{z}_{i,y}^2$$

be the corresponding values computed by the verifier. Note we do not assume knowledge of x (though, as in Section 3.1, we do assume a DDH oracle in order to identify accepting proofs).

We first show that for $j = 0, \ldots, \ell - 1$ we can compute s_j, i_j with $\mathsf{com}_j = h_3^{s_j} h_2^{i_j}$ and $i_j \in \{0, 1\}$; this implies we can also compute s^*, i with $\mathsf{com} \stackrel{\text{def}}{=} \prod_{j=0}^{\ell-1} \mathsf{com}_j^{2^j} = h_3^{s^*} h_2^i$ and $i < 2^\ell = B$. To see this, let $C_{j,0} = \mathsf{com}_j$ and $C_{j,1} = \mathsf{com}_j / h_2$. Let $\gamma_{j,1} = \gamma - \gamma_{j,0}$ and $\bar{\gamma}_{j,1} = \bar{\gamma} - \bar{\gamma}_{j,0}$ Note we must have either $\gamma_{j,0} \neq \bar{\gamma}_{j,0}$ or $\gamma_{j,1} \neq \bar{\gamma}_{j,1}$ (or possibly both). If $\gamma_{j,0} \neq \bar{\gamma}_{j,0}$ then

$$h_3^{z_{j,0}} C_{j,0}^{-\gamma_{j,0}} = C'_{j,0} = h_3^{\bar{z}_{j,0}} C_{j,0}^{-\bar{\gamma}_{j,0}}$$

and hence $\mathsf{com}_j = C_{j,0} = h_3^{(z_{j,0}-\bar{z}_{j,0})/(\gamma_{j,0}-\bar{\gamma}_{j,0})}$. If $\gamma_{j,1} \neq \bar{\gamma}_{j,1}$ then

$$h_3^{z_{j,1}} C_{j,1}^{-\gamma_{j,1}} = C'_{j,1} = h_3^{\bar{z}_{j,1}} C_{j,1}^{-\bar{\gamma}_{j,1}}$$

and hence $\mathsf{com}_j = C_{j,1} \cdot h_2 = h_3^{(z_{j,1}-\bar{z}_{j,1})/(\gamma_{j,1}-\bar{\gamma}_{j,1})} \cdot h_2$. (If both $\gamma_{j,0} \neq \bar{\gamma}_{j,0}$ and $\gamma_{j,1} \neq \bar{\gamma}_{j,1}$ then we could compute $\log_{h_2} h_3$, so we can assume this occurs with negligible probability.)

Since $\bar{\gamma} > \gamma \geq 0$ is non-zero and $A_1 = (A')^{\bar{z}_e} \bar{B}^{\bar{z}_{r_2}} \bar{A}^{-\bar{\gamma}}$, we can compute $\bar{A} = (A')^x$. We have

$$(A')^{z_e} \bar{B}^{z_{r_2}} \bar{A}^{-\gamma} = A_1 = (A')^{\bar{z}_e} \bar{B}^{\bar{z}_{r_2}} \bar{A}^{-\bar{\gamma}},$$

and hence we can compute δ_e, δ_{r_2} such that

$$\bar{A} = (A')^x = (A')^{\delta_e} \bar{B}^{\delta_{r_2}}. \tag{1}$$

We assume $\delta_{r_2} \neq 0$ (if not, then $\delta_e = \log_g w$ and we have computed the discrete logarithm of the issuer's public key). We also have

$$\bar{B}^{z_{r_3}} h_1^{z_\Delta} h_2^{z_k} h_3^{z_s} \cdot H_1^{-\gamma} = A_2 = \bar{B}^{\bar{z}_{r_3}} h_1^{\bar{z}_\Delta} h_2^{\bar{z}_k} h_3^{\bar{z}_s} \cdot H_1^{-\bar{\gamma}},$$

and hence, letting $\Delta = (z_\Delta - \bar{z}_\Delta)/(\gamma - \bar{\gamma})$ and $\delta_k = (z_k - \bar{z}_k)/(\gamma - \bar{\gamma})$,

$$H_1 = \bar{B}^{\delta_{r_3}} h_1^{\Delta} h_2^{\delta_k} h_3^{\delta_s} \tag{2}$$

for some δ_{r_3}, δ_s we can compute. For future reference, we remark that a similar calculation gives $H_1 = \bar{B}^{\delta'_{r_3}} h_1^{\Delta'} h_2^{\delta'_k} h_3^{\delta'_s}$ with $\Delta' = (z_\Delta - \hat{z}_\Delta)/(\gamma - \hat{\gamma})$ and $\delta'_{r_3}, \delta'_k, \delta'_s$ being values we can compute; the discrete-logarithm assumption implies

$$(z_\Delta - \bar{z}_\Delta)/(\gamma - \bar{\gamma}) = \Delta = \Delta' = (z_\Delta - \hat{z}_\Delta)/(\gamma - \hat{\gamma}). \tag{3}$$

Since $H_1 = g h_1^{T'} \mathsf{com}^{-1} = g h_1^{T'} h_2^{-i} h_3^{-s^*}$, substituting into Equation (2) gives

$$\bar{B}^{\delta_{r_3}} = g h_1^{T'-\Delta} h_2^{-\delta_k - i} h_3^{-\delta_s - s^*}. \tag{4}$$

Equations (1) and (4) imply that

$$(A')^{\delta_{r_3}/\delta_{r_2}} = \left(g h_1^{T'-\Delta} h_2^{-\delta_k - i} h_3^{-\delta_s - s^*} \right)^{1/(x-\delta_e)},$$

i.e., $((A')^{\delta_{r_3}/\delta_{r_2}},\ -\delta_e)$ is a signature on the vector $(T' - \Delta, -\delta_k - i, -\delta_s - s^*)$. Since signatures are only issued on[3] two-dimensional vectors, we have $-\delta_s - s^* = 0$ except with negligible probability. Letting $T = T' - \Delta$ and $k = -\delta_k - i$, we further know that this signature must correspond to an issued signature for time period T and PRF key k. Showing that $T \leq T'$ is equivalent to showing that $\Delta \geq 0$ (which we do below).

Continuing, we have

$$Y^{-z_k} (h_2 / Y^{2^\ell E})^{-\gamma} = Y_1 = Y^{-\bar{z}_k} (h_2 / Y^{2^\ell E})^{-\bar{\gamma}},$$

[3]Here we implicitly rely on the fact that the client proves knowledge of $\log_{h_2} K$ during credential generation.

and hence $Y^{-\delta_k} = h_2/Y^{2^\ell E}$ or equivalently (recalling that $-\delta_k = k+i$) $Y = h_2^{1/(k+2^\ell E+i)}$. That is, Y is indeed equal to the correct PRF value relative to the extracted signature, the epoch number E, and the index i defined by the commitments.

What remains is to use the range proof to show $\Delta \geq 0$. We have

$$D_y = C_y^{-\gamma} g^{t_y} \prod h_i^{z_{i,y}} = C_y^{-\bar{\gamma}} g^{\bar{t}_y} \prod h_i^{\bar{z}_{i,y}} = C_y^{-\hat{\gamma}} g^{\hat{t}_y} \prod h_i^{\hat{z}_{i,y}},$$

and so, letting $y_i = (z_{i,y} - \bar{z}_{i,y})/(\gamma - \bar{\gamma})$ and $y'_i = (z_{i,y} - \hat{z}_{i,y})/(\gamma - \hat{\gamma})$ for all i,

$$C_y = g^{\delta_y} \prod h_i^{y_i} = g^{\delta'_y} \prod h_i^{y'_i}$$

for some δ_y, δ'_y that we can compute. We may assume $y_i = y'_i$ for all i by the hardness of the discrete-logarithm problem. Proceeding similarly using D_*, C_*, we conclude that

$$(f_* - \bar{f}_*)/(\gamma - \bar{\gamma}) = (f_* - \hat{f}_*)/(\gamma - \hat{\gamma}). \tag{5}$$

Define $m \stackrel{\text{def}}{=} z_\Delta - \gamma\Delta$. Note that

$$\begin{aligned} m = z_\Delta - \gamma\Delta &= z_\Delta - \bar{z}_\Delta + \bar{z}_\Delta - \gamma\Delta \\ &= (\gamma - \bar{\gamma}) \cdot \Delta + \bar{z} - \gamma\Delta = \bar{z} - \bar{\gamma}\Delta \end{aligned}$$

(using Equation (3)), and similarly $m = \hat{z} - \hat{\gamma}\Delta$. Defining $m_i \stackrel{\text{def}}{=} z_{i,y} - \gamma y_i$ we similarly obtain

$$m_i = z_{i,y} - \gamma y_i = \bar{z}_{i,y} - \bar{\gamma} y_i = \hat{z}_{i,y} - \hat{\gamma} y_i.$$

We have

$$\begin{aligned} f_* &= \gamma \cdot (m + \gamma\Delta) - \sum (m_i + \gamma y_i)^2 = \gamma^2 \cdot (\Delta - \sum y_i^2) + \gamma \cdot (m - \sum m_i y_i) - \sum_i m_i^2 \\ \bar{f}_* &= \bar{\gamma} \cdot (m + \bar{\gamma}\Delta) - \sum (m_i + \bar{\gamma} y_i)^2 = \bar{\gamma}^2 \cdot (\Delta - \sum y_i^2) + \bar{\gamma} \cdot (m - \sum m_i y_i) - \sum_i m_i^2 \\ \hat{f}_* &= \hat{\gamma} \cdot (m + \hat{\gamma}\Delta) - \sum (m_i + \hat{\gamma} y_i)^2 = \hat{\gamma}^2 \cdot (\Delta - \sum y_i^2) + \hat{\gamma} \cdot (m - \sum m_i y_i) - \sum_i m_i^2. \end{aligned}$$

Thus,

$$\begin{aligned} (f_* - \bar{f}_*)/(\gamma - \bar{\gamma}) &= (\gamma + \bar{\gamma}) \cdot (\Delta - \sum y_i^2) + (m - \sum m_i y_i) \\ (f_* - \hat{f}_*)/(\gamma - \hat{\gamma}) &= (\gamma + \hat{\gamma}) \cdot (\Delta - \sum y_i^2) + (m - \sum m_i y_i). \end{aligned}$$

Since the above are equal (cf. Equation (5)), this implies $\Delta - \sum y_i^2 = 0 \bmod q$. (All equalities until now were also mod q, but now we stress it.) Since $y_i = (z_{i,y} - \bar{z}_{i,y})/(\gamma - \bar{\gamma})$, this means

$$\Delta \cdot (\gamma - \bar{\gamma})^2 - \sum (z_{i,y} - \bar{z}_{i,y})^2 = 0 \bmod q. \tag{6}$$

But

$$\begin{aligned} \left| \Delta \cdot (\gamma - \bar{\gamma})^2 - \sum (z_{i,y} - \bar{z}_{i,y})^2 \right| &\leq |T' - T| \cdot |(\gamma - \bar{\gamma})|^2 + \sum |(z_{i,y} - \bar{z}_{i,y})|^2 \\ &\leq T^* \cdot C^2 + 4 \cdot (\sqrt{T^*}\, C(L+1))^2 \\ &= T^* C^2 \cdot (1 + 4 \cdot (L+1)^2), \end{aligned}$$

where we let $T^* \geq T, T'$ be a bound on the largest time period used by the system. So long as the above is strictly smaller than q, Equation (6) must hold over the integers. But then clearly $\Delta \geq 0$

5.2 "Zero Knowledge"

It is unclear how to prove that the client's proof is zero knowledge. Instead, we suffice with sketching a proof of anonymity. That is, we show that a malicious issuer cannot distinguish a proof given by a client who holds a credential for $T_0 \leq T'$ and uses index $i_0 < B$ (with respect to committed PRF key K_0) from a proof given by a client who holds a credential for $T_1 \leq T'$ and uses index $i_1 < B$ (with respect to committed PRF key K_1), even if we allow the issuer to choose T', E, B.

Formally, fix some adversary $\mathcal{A}$ and let Real_0 correspond to the following experiment: random generators $h_1, \ldots, h_4$ are chosen and given to $\mathcal{A}$, and then $\mathcal{A}$ outputs a public key w. Next:

1. $\mathcal{A}$ is given uniform $K_0 = h_2^{k_0}$ and $K_1 = h_2^{k_1}$.
2. $\mathcal{A}$ outputs (T_0, A_0, e_0) and (T_1, A_1, e_0) along with a proof that the credentials in each case were computed correctly.
3. $\mathcal{A}$ outputs T', E, B, i_0, i_1, B with $T_0, T_1 \leq T'$ and $i_0, i_1 < B$.
4. $\mathcal{A}$ is given a proof π (relative to T', E, B) computed using credential (A_0, e_0, k_0) for T_0 and index i_0.

The final output is the view of $\mathcal{A}$, which consists of its randomness along with $h_1, \ldots, h_4$, K_0, K_1 and the proof π.

Next we consider an intermediate experiment Expt_0. Here, we first extract $x = \log_g w$ from the issuer's proof(s) of correctness during issuance. We then compute $\mathsf{to\_send}_1$ as in Real_0, but simulate $\mathsf{to\_hash}$, the challenge γ, and $\mathsf{to\_send}_2$ in the usual way, using knowledge of x to do so. The main subtlety here is to argue that the distribution of the $\{z_{i,y}\}$ is statistically close in this experiment and the previous one; the statistical difference depends on the parameter L (details omitted).

We next consider an experiment Expt' where $\mathsf{to\_hash}$, γ, and $\mathsf{to\_send}_2$ are simulated as before, but now all components of $\mathsf{to\_send}_1$ are chosen as uniform group elements. One can show that this experiment is computationally indistinguishable from Expt_0.

A symmetric argument involving experiments Expt_1 and Real_1 completes the proof.

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.