Pseudonymous Credentials for FedCM

Jonathan Katz, Samuel Schlesinger

Abstract

We describe a scheme for pseudonymous credentials.

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.

2 Definitions

In our system, we have three types of roles: *issuers*, *clients*, and *verifiers*. We start with functional definitions of *pseudonymous credentials*, followed by definitions of security.

Definition 1. *A* pseudonymous credential *consists of algorithms* (KeyGen, CredGen, ProofGen, VrfyProof) *with the following syntax:*

- KeyGen *is run by an issuer to generate keys* (pk, sk)*.*
- CredGen *is run by an issuer. At the end of the protocol, the client outputs a credential* cred*. If* cred $=\perp$ *it means the client aborted since it detected cheating.*
- ProofGen*, run by a client, takes as input a credential* cred *and a relying party identity I. It outputs a proof π.*
- VrfyProof *takes as input a public key* pk*, an identity I, and a proof π; it outputs a bit b and a pseudonym* nym*.*

Correctness requires that if a credential associated with T is used to generate a proof for honestly generated proofs verify, and different credentials result in different pseudonyms for a given relying party. (That the same credential results in the same pseudonym for a given relying party will be defined as a security requirement.)

Definition 2. *A pseudonymous credential 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 to* $\mathsf{CredGen}_{\mathsf{sk}}()$*, generate a credential* cred_i *and give it to* $\mathcal{A}$*. If* $\mathsf{cred}_i = \perp$*, set* $\mathsf{flag} := 1$.
- *On query* $\mathsf{ProofGen}(i, I)$*, run* $\pi \leftarrow \mathsf{ProofGen}(\mathsf{cred}_i, I)$ *and give* π *to* $\mathcal{A}$*. Following this, run* $(b, \mathsf{nym}) \leftarrow \mathsf{VrfyProof}_{\mathsf{pk}}(I, \pi)$ *and store the record* (i, I, nym)*. If* $b = 0$ *or there is an existing record* (j, I, nym) *with* $i \neq j$*, then set* $\mathsf{flag} := 1$.

We consider three security requirements: (1) *soundness* means that if a client has never received a credential, then it cannot generate a correct proof; (2) *unlinkability/pseudonymity* means that verifiers should not be able to link clients across different relying parties; (3) *linkability* means that a client cannot prevent being linked at a single relying party. In all cases, we assume an honest issuer who does not collude with clients or verifiers.

Definition 3 (Soundness). *A pseudonymous credential 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. $\mathcal{A}$ *outputs* I, π*, and succeeds if* $\mathsf{VrfyProof}_{\mathsf{pk}}(I, \pi) = (1, \star)$*.*

Definition 4 (Pseudonymity). *A pseudonymous credential is* pseudonymous *if the success probability of any* PPT *adversary* $\mathcal{A}$ *in the following experiment is close to* $1/2$*:*

1. *Generate keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$*, and give* pk *to* $\mathcal{A}$*. Also run* $\mathsf{CredGen}$ *twice to generate* $\mathsf{cred}_0, \mathsf{cred}_1$*.*
2. $\mathcal{A}$ *can interact with oracles* $\mathsf{ProofGen}(\mathsf{cred}_0, \cdot)$ *and* $\mathsf{ProofGen}(\mathsf{cred}_1, \cdot)$*.*
3. *At some point,* $\mathcal{A}$ *outputs* I *such that* $\mathcal{A}$ *never previously queried* $\mathsf{ProofGen}(\mathsf{cred}_0, I)$ *or* $\mathsf{ProofGen}(\mathsf{cred}_1, I)$*. In response, a uniform bit* b *is chosen, and* $\mathcal{A}$ *is given* $\mathsf{ProofGen}(\mathsf{cred}_b, I)$*.*
4. $\mathcal{A}$ *can continue to interact with oracles* $\mathsf{ProofGen}(\mathsf{cred}_0, \cdot)$ *and* $\mathsf{ProofGen}(\mathsf{cred}_1, \cdot)$*, but it may not query* $\mathsf{ProofGen}(\mathsf{cred}_0, I)$ *or* $\mathsf{ProofGen}(\mathsf{cred}_1, I)$*.*
5. *At the end of its execution,* $\mathcal{A}$ *outputs a bit* b' *and succeeds if* $b' = b$*.*

Definition 5 (Linkability). *A pseudonymous credential* linkable *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* I*, and* $\{\pi_i\}_{i=1}^{\ell+1}$*; let* $(b_i, \mathsf{nym}_i) = \mathsf{VrfyProof}_{\mathsf{pk}}(I, \pi_i)$*.* $\mathcal{A}$ *succeeds if (1)* $b_i = 1$ *for all* i *and (2) the* $\{\mathsf{nym}_i\}$ *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].

Fix a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and let $g, h_1, \ldots, h_\ell \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be (random) generators. BBS signatures work as follows:

- To generate a secret key, choose $x \leftarrow \mathbb{Z}_q$. The associated public key is $w = g_2^x$.
- 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 signature $\left(\left(g \cdot \prod_{i=1}^{\ell} h_i^{m_i}\right)^{1/(e+x)}, e\right)$.
- To verify signature (A, e) on message $(m_1, \ldots, m_\ell) \in \mathbb{Z}_q^\ell$ using public key w, check if
$$\hat{e}(A, w) = \hat{e}\left(A^{-e} g \prod h_i^{m_i}, g_2\right).$$

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 a verifier. In fact, even more is possible. Let $[\ell] = \{1, \ldots, \ell\}$, and $\mathcal{D} \subseteq [\ell]$. Then one can prove to the verifier 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 $A' := A^{r_1 r_2}$, $\bar{B} := B^{r_1}$, $\bar{A} = (A')^{-e} \cdot \bar{B}^{r_2}$, and $r_3 := r_1^{-1}$.
2. It then sends $\{m_i\}_{i \in \mathcal{D}}$ and $A', \bar{B}, \bar{A}$ along with a proof that there exist $\{m_i\}_{i \notin \mathcal{D}}, e, r_2, r_3$ such that (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 verifier $A_1 := (A')^{e'} \cdot \bar{B}^{r_2'}$ and $A_2 := \bar{B}^{r_3'} \cdot \prod_{i \notin D} h_i^{m_i'}$.
    (b) The verifier 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 verifier rejects if $A' = 1$ or $\hat{e}(A', w) \neq \hat{e}(\bar{A}, g_2)$. Otherwise, the verifier computes $H_1 := g \cdot \prod_{i \in \mathcal{D}} h_i^{m_i}$, and accepts iff $(A')^{\bar{e}} \bar{B}^{\bar{r}_2} = \bar{A}^c \cdot A_1$ and $\bar{B}^{\bar{r}_3} \prod_{i \notin \mathcal{D}} h_i^{\bar{m}_i} = H_1^c \cdot A_2$.

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

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

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

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

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

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. 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 Pseudonymous Credential

System-wide setup. Assume parameters for a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ have been established. Let $g \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators. Also fix (random) generator $h \in \mathbb{G}_1$. (This could be generated in an appropriate way using a hash function.)

Key generation. An issuer chooses a secret key $x \leftarrow \mathbb{Z}_q$; the associated public key is $w := g_2^x$.

Credential generation. To generate a credential, the issuer chooses $k, e \leftarrow \mathbb{Z}_q$ and sends $(A, e, k) = \left(\left(g \cdot h^k\right)^{1/(e+x)}, e, k\right)$ to the client. The client can verify correctness of the credential by checking that $\hat{e}(A, w) = \hat{e}(A^{-e}gh^k, g_2)$.

Proof generation. Given credential (A, e, k) and identity I, the client generates a proof as follows:

- (PoK of credential.)
  1. Choose $r_1, r_2, e', r_2', r_3', k' \leftarrow \mathbb{Z}_q$. Set $B := gh^k$.
  2. Set $A' := A^{r_1 r_2}$, $\bar{B} := B^{r_1}$, $\bar{A} := (A')^{-e}\bar{B}^{r_2}$, and $r_3 := r_1^{-1}$. Set $\mathsf{to\_send}_1 := A' \| \bar{B} \| \bar{A}$.
  3. Compute $A_1 := (A')^{e'} \cdot \bar{B}^{r_2'}$ and $A_2 := \bar{B}^{r_3'}h^{k'}$, and set $\mathsf{to\_hash} := A_1 \| A_2$.
- (PRF + associated proof.)
  1. Set $Y := g^{1/(k+I)}$, and $\mathsf{to\_send}_1 := \mathsf{to\_send}_1 \| Y$.
  2. Set $Y_1 := Y^{-k'}$ and $\mathsf{to\_hash} := \mathsf{to\_hash} \| Y_1$.
- Compute $\gamma := H(\mathsf{to\_send}_1 \| \mathsf{to\_hash})$ for H a suitable hash function with range $\mathbb{Z}_q$.
- (Complete PoKs.) Compute $z_e := -\gamma e + e'$, $z_{r_2} := \gamma \cdot r_2 + r_2'$, $z_{r_3} := \gamma \cdot r_3 + r_3'$, and $z_k := -\gamma \cdot k + k'$. Set $\mathsf{to\_send}_2 := z_e \| z_{r_2} \| z_{r_3} \| z_k$.
- The final proof is $\mathsf{to\_send}_1, \gamma, \mathsf{to\_send}_2$.

Verification. The verifier holds I and a public key w, and receives a proof $\mathsf{to\_send}_1, \gamma, \mathsf{to\_send}_2$. It begins by parsing $\mathsf{to\_send}_1$ as $A' \| \bar{A} \| \bar{B} \| Y$ and $\mathsf{to\_send}_2$ as $z_e \| z_{r_2} \| z_{r_3} \| z_k$. It first checks that $A' \neq 1$ and $\hat{e}(A', w) = \hat{e}(\bar{A}, g_2)$, rejecting if these do not hold. Otherwise, it does:

- (Verify PoK of credential.) It computes
$$A_1 := (A')^{z_e} \bar{B}^{z_{r_2}} \bar{A}^{-\gamma} \quad \text{and} \quad A_2 := \bar{B}^{z_{r_3}} h^{z_k} g^{-\gamma}.$$
It then sets $\mathsf{to\_hash} := A_1 \| A_2$.
- (Verify PRF proof.) Set $Y_1 := Y^{-z_k} \cdot (g/Y^I)^{-\gamma}$ and $\mathsf{to\_hash} := \mathsf{to\_hash} \| Y_1$.
- Finally, it accepts iff $H(\mathsf{to\_send}_1 \| \mathsf{to\_hash}) \stackrel{?}{=} \gamma$.

Correctness. Correctness is left as an exercise for the implementer! But we also verify correctness by showing that corresponding elements in $\mathsf{to\_hash}$ computed by the prover and verifier are equal.

- As in Section 3.1, we have $\hat{e}(A', w) = \hat{e}(\bar{A}, g_2)$. The client computes $A_1 = (A')^{e'} \cdot \bar{B}^{r'_2}$. The verifier 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 $A_2 = \bar{B}^{r'_3} h^{k'}$. The issuer computes
$$\begin{aligned} A_2 &= \bar{B}^{z_{r_3}} h^{z_k} g^{-\gamma} \\ &= \left(\bar{B}^{r_3} h^{-k}\right)^{\gamma} \bar{B}^{r'_3} h^{k'} g^{-\gamma} \\ &= \bar{B}^{r'_3} h^{k'}. \end{aligned}$$
- The client computes $Y_1 = Y^{-k'}$. The issuer computes
$$\begin{aligned} Y_1 = Y^{-z_k} \cdot (g/Y^I)^{-\gamma} &= Y^{\gamma k} Y^{-k'} (g/Y^I)^{-\gamma} \\ &= \left(Y^{k+I}\right)^{\gamma} Y^{-k'} g^{-\gamma} \quad = \quad Y^{-k'}. \end{aligned}$$

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`.