Block-wise P-Signatures and Non-Interactive Anonymous Credentials with Efficient Attributes

Abstract. Anonymous credentials are protocols in which users obtain certificates from organizations and subsequently demonstrate their possession in such a way that transactions carried out by the same user cannot be linked. We present an anonymous credential scheme with non-interactive proofs of credential possession where credentials are associated with a number of attributes. Following recent results of Camenisch and Groß (CCS 2008), the proof simultaneously convinces the verifier that certified attributes satisfy a certain predicate. Our construction relies on a new kind of P-signature, termed *block-wise P-signature*, that allows a user to obtain a signature on a committed vector of messages and makes it possible to generate a short witness that serves as a proof that the signed vector satisfies the predicate. A non-interactive anonymous credential is obtained by combining our *block-wise* P-signature scheme with the Groth-Sahai proof systems. It allows efficiently proving possession of a credential while simultaneously demonstrating that underlying attributes satisfy a predicate corresponding to the evaluation of inner products (and therefore disjunctions or polynomial evaluations). The security of our scheme is proved in the standard model under non-interactive assumptions.

Keywords. P-signatures, anonymous credentials, anonymity, non-interactive proofs, standard model, efficient attributes.

1 Introduction

Introduced by Chaum [22] and extensively studied in the last two decades [26, 12, 40, 18, 19, 21, 20, 3, 4], anonymous credential systems enable users to authenticate themselves in a privacy-preserving manner. In such a protocol, a user can prove that an organization has supplied him with a certificate in such a way that the request for a certificate cannot be linked to any of its proofs of possession and multiple proofs involving the same credential cannot be linked to each other. In many realistic applications, it is desirable to augment digital credentials with a number of user attributes (such as their citizenship, their birth date, their obtained degrees, . . .) while allowing users to selectively disclose some of their attributes or efficiently prove properties about them without disclosing any other information. This problem was addressed by Camenisch and Groß [15] who showed how to conveniently extend the Camenisch-Lysyanskaya construction [18, 19] into an anonymous credential system with efficient attributes. In this paper, we consider similar problems in the context of *non-interactive* anonymous credentials in the standard model, as formalized in [3].

Anonymous credential systems usually combine two essential components. The first one is a protocol allowing a user to obtain a signature from an organization on a committed value (which is typically the user's private key) by sending a commitment to the signer and eventually obtaining a signature on the message without leaking useful information on the latter. The second component is a proof of knowledge of a signature on a committed value. Namely, the prover holds a pair (m, σ), reveals a commitment c to m and demonstrates his possession of σ as a valid signature on m.

Prior Work. While the above tasks can always be inefficiently realized by combining general two-party computation protocols [47] and zero-knowledge proofs for any NP statements [32], Camenisch and Lysyanskaya [18, 19] used groups of hidden order and Fujisaki-Okamoto commitments [30] to build the first practical realizations 10 years ago. Their approach was subsequently extended to groups of public order using bilinear maps [20, 2].

Until recently, all anonymous credential systems required users to engage in an interactive conversation with the verifier to convince him of their possession of a credential. While interaction can be removed using the Fiat-Shamir paradigm [27] and the random oracle model [6], this methodology is limited (see [23, 33] for instance) to only give heuristic arguments in terms of security. This motivated Belenkiy, Chase, Kohlweiss and Lysyanskaya [3] to design non-interactive[1] anonymous credentials in the standard model – assuming a common reference string – using an underlying primitive named *P-signature* (as a shorthand for signatures with efficient Protocols). Their results were extended by [5] (and, more recently, in [29]) into non-interactive anonymous credential schemes supporting credential delegation.

Credentials Supporting Efficient Attributes. In real-life situations, users holding a number of certified attributes may be willing to selectively disclose a restricted number of their attributes while preserving their privacy and the secrecy of their other attributes. A natural approach is to extend classical anonymous credentials such as [18, 20] using generalizations of the Pedersen commitment [42] allowing to commit to n attributes at once in groups of hidden order. However, disclosing a single specific attribute entails to commit to $n-1$ attributes so as to prove that one attribute matches the disclosed value and committed attributes are the remaining certified ones. The drawback of this technique is that each proof has linear size in the overall number of attributes.

To address this concern, Camenisch and Groß [15] suggested a completely different technique consisting in encoding attributes as prime numbers. Basically, users first obtain a signature on two committed messages: the first one is the user's private key and the second one consists of the product of all users' attributes. Later on, when the user wants to prove his ownership of a credential containing a certain attribute, he just has to prove that this attribute divides the second committed message. Camenisch and Groß also showed how users can prove that they hold an attribute appearing in some public attribute list and how to handle negated statements (namely, prove that a certain attribute is not contained in their attribute set). They also showed how to extend their techniques and prove the conjunction or the disjunction of simple such atomic statements. Unfortunately, their techniques cannot be applied in the setting of non-interactive anonymous credentials as they inherently rely on groups of hidden order, which makes them hardly compatible with the Groth-Sahai proof systems [34] used in [3, 5]. It turns out that efficiently handling attributes in this context requires new techniques to be worked out.

In [45], Shahandashti and Safavi-Naini used threshold attribute-based signatures [41] to construct attribute-based anonymous credentials where users can prove threshold predicates (*i.e.*, the ownership of t-out-of-n public attributes). However, their construction requires interaction and is not meant to provide compact proofs, which is the focus of this paper.

Our Contribution. This paper presents an anonymous credential scheme allowing to non-interactively prove the possession of a credential associated with attributes that satisfy a given predicate without leaking any further information. To this end, we extend the approach of [3] by introducing a new kind of P-signature termed *block-wise* P-signature. In a nutshell, such a primitive is a P-signature allowing a user to obtain a signature on a committed vector of messages (similarly to the multi-block P-signature of [5]). Unlike [5] however, our P-signature makes it possible for the user to generate a short NIZK argument (*i.e.*, the size of which does not depend on the vector size) that serves as evidence that the signed vector satisfies a certain predicate.

[1] The protocol for obtaining a signature on a committed message still demands interaction but the proving phase, which is usually more frequently executed, consists of one message from the prover to the verifier.

Inspired by the work of Katz, Sahai, Waters [38], we present a block-wise P-signature for predicates corresponding to the zero or non-zero evaluation of inner products (and therefore disjunctions or polynomial evaluations). By combining our block-wise P-signature with the Groth-Sahai methodology [34] as in [3], we readily obtain an efficient non-interactive anonymous credential supporting efficient attributes. By appropriately using the inner product with suitable attribute encodings, we notably obtain (1) an efficient way for users to prove that specific attributes appear in their attribute set; (2) a method for concisely proving the inclusion of one of their attributes in a public list; (3) short proofs that the certified attribute set contains a certain (exact or inexact) threshold of binary attributes (in a similar way, we can prove that a subset of the certified set is at most t binary attributes away from some public attribute set). Using a very small amount of interaction (namely, verifiers just have to send a challenge consisting of a short random value in $\mathbb{Z}_p$, where p is the group order), we can also handle conjunctions of atomic conditions and even more complex formulas such as CNF or DNF in two rounds. The non-interactivity property is unfortunately lost when we want to deal with CNF/DNF formulas but our solution still decreases the number of rounds w.r.t. traditional interactive constructions. Indeed, at least 3 rounds are needed in interactive proofs using Σ protocols.

The security of our scheme is proved in the standard model under non-interactive assumptions. Although our scheme does not perform as well as the Camenisch-Groß system (notably because, unlike [15], we cannot prevent the public key size from depending on the number n of attributes), this yields the first result on non-interactive anonymous credentials with efficient attributes in the standard model. Like [3, 5], we rely on a common reference string and only need interaction in the protocol allowing users to obtain their credentials (except for predicates involving conjunctions).

Organization. In section 2, we first give formal definitions of block-wise F-unforgeable signatures (similarly to [3], we can only prove a relaxed form of unforgeability which suffices in this context) and block-wise P-signatures. Our realization for inner product relations is described in section 3. Its application to the realization of anonymous credentials with efficient attributes is detailed in appendix E, where we also discuss the efficiency of the scheme and the kind of predicates that can be expressed using inner products.

2 Background and Definitions

Notations. We say that a function $\nu : \mathbb{N} \rightarrow [0, 1[$ is negligible if for, any polynomial $p(.)$, we have $\nu(\lambda) < |1/p(\lambda)|$ for any sufficiently large $\lambda \in \mathbb{N}$. If $A(x) \leftrightarrows B(y)$ denotes an interactive protocol between A and B on input x and y, respectively, and if participant A (resp. B) outputs a bit $b \in \{0, 1\}$ after the execution of the protocol, we write $b \Leftarrow A(x) \leftrightarrows B(y)$ (resp. $A(x) \leftrightarrows B(y) \Rightarrow b$).

2.1 Bilinear Maps and Complexity Assumptions

We consider bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p with a mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that: (1) $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$; (2) $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

Definition 1 ([9]). *In a group $\mathbb{G}$ of prime order p, the* **Decision Linear Problem** *(DLIN) is to distinguish the distributions $(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ and $(g, g^a, g^b, g^{ac}, g^{bd}, g^z)$, with $a, b, c, d, z \overset{R}{\leftarrow} \mathbb{Z}_p$. The* **Decision Linear Assumption** *is the intractability of DLIN for any PPT distinguisher $\mathcal{D}$.*

This problem is to decide if vectors $\vec{g_1} = (g^a, 1, g)$, $\vec{g_2} = (1, g^b, g)$ and $\vec{g_3}$ are linearly dependent.

Like several previous P-signatures, our scheme uses the Hidden Strong Diffie-Hellman assumption [11] that strengthens a "q-type" assumption introduced in [8].

Definition 2 ([11]). *The* **q-Hidden Strong Diffie-Hellman problem** *(q-HSDH) consists in, given $(g, u, \Omega = g^{\omega}) \in \mathbb{G}^3$ and a set of q tuples $(g^{1/(\omega+c_i)}, g^{c_i}, u^{c_i})$ with $c_1, \ldots, c_q \xleftarrow{R} \mathbb{Z}_p^*$, finding $(g^{1/(\omega+c)}, g^c, u^c)$ such that $c \neq c_i$ for $i = 1, \ldots, q$.*

We also use the following problem, which is not easier than the problem, used in [37], of finding a pair $(g^{\mu}, g^{\mu ab}) \in (\mathbb{G}\backslash\{1_{\mathbb{G}}\})^2$ given $(g, g^a, g^b) \in \mathbb{G}^3$.

Definition 3. *The* **Flexible Diffie-Hellman problem** *(FlexDH) in $\mathbb{G}$ is, given $(g, g^a, g^b) \in \mathbb{G}^3$, where $a, b \xleftarrow{R} \mathbb{Z}_p^*$, to find a triple $(g^{\mu}, g^{\mu a}, g^{\mu ab})$ such that $\mu \neq 0$.*

The paper will make use of two other problems. The first one was introduced – in a potentially easier variant – in [10].

Definition 4 ([10]). *Let $\mathbb{G}$ be a group of prime order p. The* **n-Diffie-Hellman Exponent** *(n-DHE) problem is, given elements $(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n}$ such that $g_i = g^{(\alpha^i)}$ for each $i \in [1, 2n]\backslash\{n+1\}$ and where $\alpha \xleftarrow{R} \mathbb{Z}_p^*$, to compute the missing element $g_{n+1} = g^{(\alpha^{n+1})}$.*

We finally need an assumption that strengthens the n-DHE assumption in the same way as the FlexDH assumption is a strengthening of the Diffie-Hellman assumption.

Definition 5. *Let $\mathbb{G}$ be a group of prime order p. The* **Flexible n-Diffie-Hellman Exponent** *(n-FlexDHE) problem is, given elements $(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n}$ such that $g_i = g^{(\alpha^i)}$ for each $i \in [1, 2n]\backslash\{n+1\}$ and where $\alpha \xleftarrow{R} \mathbb{Z}_p^*$, to compute a non-trivial triple $(g^{\mu}, g_{n+1}^{\mu}, g_{2n}^{\mu}) \in (\mathbb{G}\backslash\{1_{\mathbb{G}}\})^3$, for some $\mu \in \mathbb{Z}_p^*$ and where $g_{n+1} = g^{(\alpha^{n+1})}$.*

Evidence of the generic intractability of the n-FlexDHE assumption is provided in appendix F.

2.2 Commitments to Vectors

We consider perfectly hiding commitments (VecCom, VecOpen) allowing to commit to vectors. In the following, we denote by $V = \text{VecCom}(\vec{m}; r)$ the result of committing to $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$ using randomness $r \in \mathbb{Z}_p$. In addition, we require that commitments be openable in a coordinate-wise manner and call $W = \text{VecOpen}(\vec{m}, r, i)$ the opening of V in position $i \in [1, n]$. Such a pairing-based Pedersen-like commitment [42], based on ideas from [10, 17], was described in [39]. The commitment key is $(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n}$ where $g_i = g^{(\alpha^i)}$ for each i. To commit to a vector $\vec{m} = (m_1, \ldots, m_n)$, the committer picks $r \xleftarrow{R} \mathbb{Z}_p$ and computes $V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$. Thanks to the specific choice of $\{g_i\}_{i \in [1,2n]\backslash\{n+1\}}$, $W_i = g_i^r \cdot \prod_{j=1, j\neq i}^{n} g_{n+1-j+i}^{m_j}$ serves as evidence that m_i is the i-th component of $\vec{m}$ as it satisfies the relation $e(g_i, V) = e(g, W_i) \cdot e(g_1, g_n)^{m_i}$. The opening $W_i = \text{VecOpen}(V, \vec{m}, r, i)$ at position i is easily seen not to reveal anything about other components of $\vec{m}$. Moreover, the infeasibility of opening a commitment to two distinct messages for some coordinate $i \in [1, n]$ relies on the n-DHE assumption.

2.3 Block-wise F-Unforgeable Signatures and P-Signatures

We begin by defining block-wise F-unforgeable signatures. As introduced in [3], F-unforgeability refers to the infeasibility for the adversary to craft a valid signature on some message $m \in Z_p$ while only outputting $F(m)$, for some injective function F, instead of m itself. The need for such a relaxation stems from the limited extractability of Groth-Sahai proofs: in a nutshell, only g^m is efficiently extractable from a commitment to $m \in \mathbb{Z}_p$ using the trapdoor of the CRS.

For reasons that will become apparent later on, block-wise F-unforgeable signatures will be associated with two families of relations that we call $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively.

Definition 6. *Let $\mathcal{D}$ be a domain and let $\mathcal{R}_1$ and $\mathcal{R}_2$ be families of efficiently computable relations such that each $R \in \mathcal{R}_1 \cup \mathcal{R}_2$ is of the form $R : [0,n] \times \mathcal{D}^n \times \mathcal{D}^n \rightarrow \{0,1\}$ for some $n \in \mathbb{N}$. A block-wise signature for $(\mathcal{R}_1, \mathcal{R}_2, \mathcal{D})$ consists of a tuple $\Sigma = (\mathsf{Setup}, \mathsf{SigSetup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Witness\text{-}Gen}, \mathsf{Witness\text{-}Verify})$ of algorithms with the following specifications.*

Setup(λ)**:** *takes as input a security parameter λ and outputs a set of public parameters* params.

SigSetup(λ, n)**:** *takes as input a security parameter $\lambda \in \mathbb{N}$ and an integer $n \in \mathsf{poly}(\lambda)$ denoting the length of message vectors to be signed. It outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$.

Sign$(\mathsf{sk}, \vec{m})$**:** *is a (possibly randomized) algorithm that takes as input a private key* sk *and a vector $\vec{m} = (m_1, \ldots, m_n)$ of messages where $m_i \in \mathcal{D}$ for $i = 1$ to n. It outputs a signature σ on $\vec{m}$.*

Verify$(\mathsf{pk}, \vec{m}, \sigma)$**:** *is a deterministic algorithm that takes as input a public key* pk*, a signature σ and a message vector $\vec{m} = (m_1, \ldots, m_n)$. It outputs 1 if σ is deemed valid for $\vec{m}$ and 0 otherwise.*

Witness-Gen$(\mathsf{pk}, R, i, \vec{m}, \vec{X}, \sigma)$**:** *takes as input a public key* pk*, a relation $R \in \mathcal{R}_1 \cup \mathcal{R}_2$, an integer $i \in [0,n]$, two distinct vectors $\vec{m} = (m_1, \ldots, m_n) \in \mathcal{D}^n$ and $\vec{X} = (x_1, \ldots, x_n) \in \mathcal{D}^n$, and a signature σ. If $\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) = 0$ or $R(i, \vec{m}, \vec{X}) = 0$, it outputs $\perp$. Otherwise, it returns a witness W proving that σ is a signature on some $\vec{m} \in \mathcal{D}^n$ s.t. $R(i, \vec{m}, \vec{X}) = 1$.*

Witness-Verify$(\mathsf{pk}, R, i, \vec{X}, W, \sigma)$**:** *is a deterministic algorithm that takes in a public key* pk*, a relation $R \in \mathcal{R}_1 \cup \mathcal{R}_2$, an integer $i \in [0,n]$, a vector $\vec{X} \in \mathcal{D}^n$, a witness W and a signature σ. It outputs 1 if W is deemed as convincing evidence that σ is a valid signature of some vector $\vec{m} = (m_1, \ldots, m_n) \in \mathcal{D}^n$ such that $R(i, \vec{m}, \vec{X}) = 1$.*

Except Setup, these algorithms all implicitly take public parameters params as additional inputs. To lighten notations, we omit to explicitly write them.

The following security definitions considers two kinds of forgers. The first one – which corresponds to case (i) in the definition – refers to attacks where the adversary outputs a new signature that was not legally obtained by invoking the signing oracle. The second one – captured by case (ii) – relates to forgeries where the adversary re-uses a signature (say σ_j for some $j \in \{1, \ldots, q\}$) that *was* produced by the signing oracle but manages to prove a property that is *not* satisfied by the signed vector $\vec{m}_j$.

In the context of case (ii), we need to consider two families of relations. The first one is called $\mathcal{R}_1$ and includes relations R_1 for which the adversary illegitimately proves that $R_1(i, \vec{m}_j, \vec{X}^\star) = 1$ and only outputs $F(\vec{X}^\star) = (F(x_1^\star), \ldots, F(x_n^\star))$. The second relation family $\mathcal{R}_2$ comprises relations R_2 for which the adversary tricks the verifier into believing that $R_2(i, \vec{m}_j, \vec{X}^\star) = 1$ and explicitly outputs $\vec{X}^\star = (x_1^\star, \ldots, x_n^\star)$ instead of $F(\vec{X}^\star)$. We cannot consider a single relation family unifying both $\mathcal{R}_1$ and $\mathcal{R}_2$ because, for technical reasons, our security proof ceases to work if the adversary only outputs $F(\vec{X}^\star)$ in the case of relations $R_2 \in \mathcal{R}_2$ (as explained by remark 1 in appendix B). At the same time, we also need to consider relations $R_1 \in \mathcal{R}_1$ because of the limited extractability properties of Groth-Sahai proofs.

In the notations of definition 7, $\varUpsilon \subset \{1, \ldots, n\}$ denotes the smallest subset such that values $\{F(x_t)\}_{t \in \varUpsilon}$ make it possible to verify that $\vec{X} = (x_1, \ldots, x_n)$ satisfies $R_1(i, \vec{m}, \vec{X}) = 1$.

Definition 7. *Let $\mathcal{R}_1, \mathcal{R}_2$ be families of relations over $[0,n] \times \mathcal{D}^n \times \mathcal{D}^n$ for some domain $\mathcal{D}$. A block-wise signature scheme Σ is said $(F, \mathcal{R}_1, \mathcal{R}_2)$-**unforgeable** for some efficiently computable injective function $F(.)$, if any PPT adversary has negligible advantage in the following game:*

1. *The challenger runs* $\mathsf{Keygen}(\lambda, n)$ *and obtains* $(\mathsf{pk}, \mathsf{sk})$ *before sending* pk *to $\mathcal{A}$.*
2. *$\mathcal{A}$ adaptively queries a signing oracle on up to $q \in \mathsf{poly}(\lambda)$ occasions. At each query $j \in [1,q]$, $\mathcal{A}$ chooses a vector $\vec{m} = (m_1, \ldots, m_n)$ and obtains $\sigma_j = \mathsf{Sign}(\mathsf{sk}, \vec{m})$.*

3. *$\mathcal{A}$ outputs a tuple $(\mathsf{Pred}^\star, W^\star, \sigma^\star)$ consisting of a predicate $\mathsf{Pred}^\star$, a witness $W^\star$ and a signature $\sigma^\star$. The predicate $\mathsf{Pred}^\star$ consists of a triple which is either of the form $(R_1, i, \{F(x_t^\star)\}_{t\in\Upsilon})$, for some subset $\Upsilon \subset \{1,\ldots,n\}$ such that $i \in \Upsilon$, or $(R_2, i, \vec{X}^\star)$ where $i \in [0,n]$ is an index, $R_1 \in \mathcal{R}_1$ and $R_2 \in \mathcal{R}_1 \cup \mathcal{R}_2$ are relations and $\vec{X}^\star = (x_1^\star, \ldots, x_n^\star) \in \mathcal{D}^n$ is a vector. The adversary wins if: (a) $\mathsf{Witness\text{-}Verify}(\mathsf{pk}, R, i, \vec{X}^\star, W^\star, \sigma^\star) = 1$. (b) It holds that either:*

   (i) $\sigma^\star$ was not the output of any signing query;

   (ii) $\sigma^\star = \sigma_j$, for some query $j \in [1,q]$, but the queried vector $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$ was such that $R_1(i, \vec{m}_j, \vec{X}^\star) = 0$ (resp. $R_2(i, \vec{m}_j, \vec{X}^\star) = 0$) while the predicate $\mathsf{Pred}^\star$ is of the form $(R_1, i, \{F(x_t^\star)\}_{t\in\Upsilon})$ (resp. $(R_2, i, \vec{X}^\star)$).

   The adversary $\mathcal{A}$'s advantage is its probability of being successful, taken over all random coins.

¿From a block-wise F-unforgeable signature, a full-fledged block-wise P-signature is obtained as specified by definition 8.

Definition 8. *A block-wise P-signature combines a $(F, \mathcal{R}_1, \mathcal{R}_2)$-unforgeable block-wise signature with a vector commitment* (VecCom, VecOpen), *a perfectly binding commitment* $(\mathsf{Com}, \mathsf{Open})$ *and:*

1. *An algorithm $\mathsf{SigProve}_1\big(\mathsf{pk}, R_1, i, \Upsilon, \sigma, \vec{m} = (m_1, \ldots, m_n), \vec{X} = (x_1, \ldots, x_n)\big)$ that, for some relation $R_1 \in \mathcal{R}_1$ and some subset $\Upsilon \subset \{1,\ldots,n\}$ such that $i \in \Upsilon$, generates commitments $\{C_{x_t}\}_{t\in\Upsilon}$, C_W, C_σ and a NIZK proof*

$$\pi \leftarrow \mathsf{NIZPK}\big(\{x_t \text{ in } C_{x_t}\}_{t\in\Upsilon},\ W \text{ in } C_W,\ \sigma \text{ in } C_\sigma \mid \{(W, \{F(x_t)\}_{t\in\Upsilon}, \sigma) :$$
$$\exists\ \vec{m}\ s.t.\ \mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) = 1\ \wedge\ \mathsf{Witness\text{-}Verify}(\mathsf{pk}, R_1, i, \vec{X}, W, \sigma) = 1\}\big),$$

   and the corresponding $\mathsf{VerifyProof}_1(\mathsf{pk}, R_1, i, \pi, C_\sigma, C_W, \{C_{x_t}\}_{t\in\Upsilon})$ algorithm.

2. *An algorithm $\mathsf{SigProve}_2(\mathsf{pk}, R, i, \sigma, \vec{m}, \vec{X})$ that, for some relation $R \in \mathcal{R}_1 \cup \mathcal{R}_2$, generates commitments C_W, C_σ and a proof*

$$\pi \leftarrow \mathsf{NIZPK}\big(W \text{ in } C_W,\ \sigma \text{ in } C_\sigma \mid \{(W, \sigma) : \exists\ \vec{m}\ s.t.\ \mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) = 1$$
$$\wedge\ \mathsf{Witness\text{-}Verify}(\mathsf{pk}, R, i, \vec{X}, W, \sigma) = 1\}\big)$$

   with its corresponding $\mathsf{VerifyProof}_2(\mathsf{pk}, R, i, \pi, C_\sigma, C_W, \vec{X})$ algorithm.

3. *A NIZK proof that two perfectly binding commitments open to the same value, i.e., an algorithm $\mathsf{EqComProve}$ outputting a proof of membership for the language*

$$L = \{(C, D)\ s.t.\ \exists\ (x, y), (open_x, open_y) \mid C = \mathsf{Com}(x, open_x)\ \wedge\ D = \mathsf{Com}(y, open_y)\ \wedge\ x = y\}.$$

4. *$\mathsf{SigIssue}\big(\mathsf{sk}, V', (m_{n_1+1}, \ldots, m_n)\big) \leftrightarrows \mathsf{SigObtain}(\mathsf{pk}, \vec{m}_{|n_1} = (m_1, \ldots, m_{n_1}), open_{\vec{m}_{|n_1}})$ is an interactive protocol allowing a user to obtain a signature σ on $\vec{m} = (m_1, \ldots, m_{n_1}, m_{n_1+1}, \ldots, m_n)$ without letting the signer – whose input consists of $V' = \mathrm{VecCom}(\vec{m}_{|n_1}, r')$, for some r' and $n_1 \in [1,n]$, and public messages $(m_{n_1+1}, \ldots, m_n)$ – learn anything about $\vec{m}_{|n_1}$.*

In this definition, $\Upsilon \subset \{1,\ldots,n\}$ is the smallest subset such that commitments $\{C_{x_t}\}_{t\in\Upsilon}$ allow verifying the proof that the underlying vector $\vec{X}$ satisfies $R_1(i, \vec{m}, \vec{X}) = 1$.

Unforgeability of P-signatures. To define the unforgeability of block-wise P-signatures, we shall assume that $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ starts with the user $\mathcal{U}$ committing to a vector $(m_1, \ldots, m_{n_1})$

and interactively proving to the issuer his knowledge of an opening of the commitment. We require the existence of a knowledge extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{SigObtain}}$ that can extract the committed vector $(m_1, \ldots, m_{n_1})$ by rewinding the prover $\mathcal{A}$. Since $(\mathrm{VecCom}, \mathrm{VecOpen})$ is a perfectly hiding commitment, this will be necessary to formalize the unforgeability of our P-signatures. We note that a similar approach was taken in [16] to define specific security properties of e-cash systems.

Definition 9. *A block-wise P-signature Σ is $(F, \mathcal{R}_1, \mathcal{R}_2)$-***unforgeable***, for relation families $\mathcal{R}_1, \mathcal{R}_2$, if there are efficient algorithms* $(\mathsf{ExtractSetup}, \mathsf{Extract})$ *s.t. (i) the output distributions of* Setup *and* $\mathsf{ExtractSetup}$ *are statistically close; (ii) any PPT algorithm $\mathcal{A}$ has negligible advantage in this game.*

1. *The challenger runs* $\mathsf{params} \leftarrow \mathsf{ExtractSetup}(\lambda)$ *and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SigSetup}(\lambda, n)$*, for some integer $n \in \mathsf{poly}(\lambda)$, and hands* pk *to $\mathcal{A}$.*
2. *On up to $q \in \mathsf{poly}(\lambda)$ occasions, $\mathcal{A}$ triggers an execution of* $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ *and acts as a user interacting with the* $\mathsf{SigIssue}$*-executing challenger. At each such execution $j \in [1, q]$, the challenger runs $\mathcal{E}^{\mathcal{A}}_{\mathsf{SigObtain}}$ so as to extract $\mathcal{A}$'s vector $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$ (or, more precisely, the restriction $(m_{j,1}, \ldots, m_{j,n_1})$ to its first n_1 coordinates, for some $n_1 \in [1, n]$) and bookkeeps it. We denote by σ_j the signature obtained by $\mathcal{A}$ at the end of the j-th execution of* $\mathsf{SigObtain}$.
3. *$\mathcal{A}$ outputs commitments C_σ, C_W, a proof π and a statement* claim *consisting of a triple which is either of the form $(R_1, i, \{C_{x_t}\}_{t \in \Upsilon})$ or $(R_2, i, \vec{X})$, for some integer $i \in [0, n]$, some relations $R_1 \in \mathcal{R}_1$ or $R_2 \in \mathcal{R}_1 \cup \mathcal{R}_2$, some vector $\vec{X} = (x_1, \ldots, x_n) \in \mathcal{D}^n$ or some commitments $\{C_{x_t}\}_{t \in \Upsilon}$ – for some subset $\Upsilon \subset \{1, \ldots, n\}$ – to elements $x_t \in \mathcal{D}$. The adversary $\mathcal{A}$ is successful if:*

   a. *Exactly one of the following conditions is satisfied.*

      1. $\mathsf{claim} = (R_1, i, \{C_{x_t}\}_{t \in \Upsilon})$ *and* $\mathsf{VerifyProof}_1(\mathsf{pk}, R_1, i, \pi, C_\sigma, C_W, \{C_{x_t}\}_{t \in \Upsilon}) = 1$.
      2. claim *is of the form $(R_2, i, \vec{X})$ and it holds that* $\mathsf{VerifyProof}_2(\mathsf{pk}, R_2, i, \pi, C_\sigma, C_W, \vec{X}) = 1$.

   b. *If we define the predicate* Pred *to be $(R, i, \{\mathsf{Extract}(C_{x_t})\}_{t \in \Upsilon})$ in situation 1 and simply* claim *in situation 2, the triple $\big(\mathsf{Pred}, \mathsf{Extract}(C_W), \mathsf{Extract}(C_\sigma)\big)$ forms a successful forgery in the game of definition 7 when the vectors $\vec{m}_1, \ldots, \vec{m}_q$ are those queried for signature.*

$\mathcal{A}$'s advantage is its success probability, taken over all coin tosses.

Belenkiy *et al.* [3] formalized other security notions named signer privacy, user privacy and zero-knowledge that P-signatures ought to satisfy. These notions are formally defined in appendix A.

2.4 Groth-Sahai Proofs

In the following notations, for equal-dimension vectors $\vec{A}$ and $\vec{B}$ containing exponents or group elements, $\vec{A} \odot \vec{B}$ stands for their component-wise product.

To simplify the description, our scheme uses Groth-Sahai proofs based on the DLIN assumption although instantiations based on the symmetric external Diffie-Hellman assumption are also possible. In the DLIN setting, the Groth-Sahai (GS) proof systems [34] use a common reference string comprising vectors $\vec{f_1}, \vec{f_2}, \vec{f_3} \in \mathbb{G}^3$, where $\vec{f_1} = (f_1, 1, g)$, $\vec{f_2} = (1, f_2, g)$ for some $f_1, f_2, g \in \mathbb{G}$. To commit to $X \in \mathbb{G}$, one sets $\vec{C} = (1, 1, X) \odot \vec{f_1}^{\,r} \odot \vec{f_2}^{\,s} \odot \vec{f_3}^{\,t}$ with $r, s, t \stackrel{R}{\leftarrow} \mathbb{Z}_p$. When proofs should be perfectly sound, $\vec{f_3}$ is set as $\vec{f_3} = \vec{f_1}^{\,\xi_1} \odot \vec{f_2}^{\,\xi_2}$ with $\xi_1, \xi_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Commitments $\vec{C} = (f_1^{r+\xi_1 t}, f_2^{s+\xi_2 t}, X \cdot g^{r+s+t(\xi_1+\xi_2)})$ are then Boneh-Boyen-Shacham (BBS) ciphertexts [9] that can be decrypted using $\alpha_1 = \log_g(f_1)$, $\alpha_2 = \log_g(f_2)$. In the witness indistinguishability (WI) setting, defining $\vec{f_3} = \vec{f_1}^{\xi_1} \odot \vec{f_2}^{\xi_2} \odot (1,1,g)^{-1}$ gives linearly independent $(\vec{f_1}, \vec{f_2}, \vec{f_3})$ and $\vec{C}$ is a perfectly hiding commitment. Under the DLIN assumption, the two settings are indistinguishable. In either case, the commitment is written $\vec{C} = \mathsf{GSCom}(X, \mathit{open}_X)$ and $\mathit{open}_X = (r,s,t)$ is its opening.

To commit to an exponent $x \in \mathbb{Z}_p$, one computes $\vec{C} = \vec{\varphi}^x \odot \vec{f_1}^r \odot \vec{f_2}^s$, with $r, s \xleftarrow{R} \mathbb{Z}_p^*$, using a CRS comprising vectors $\vec{\varphi}, \vec{f_1}, \vec{f_2}$. The commitment and its opening are denoted by $\vec{C} = \mathsf{GSCom}(x, \mathit{open}_x)$ and $\mathit{open}_x = (r,s)$, respectively. In the soundness setting $\vec{\varphi}, \vec{f_1}, \vec{f_2}$ are linearly independent vectors (typically, one chooses $\vec{\varphi} = \vec{f_3} \odot (1,1,g)$ where $\vec{f_3} = \vec{f_1}^{\xi_1} \odot \vec{f_2}^{\xi_2}$) whereas, in the WI setting, choosing $\vec{\varphi} = \vec{f_1}^{\xi_1} \odot \vec{f_2}^{\xi_2}$ gives a perfectly hiding commitment since $\vec{C}$ is always a BBS encryption of $1_{\mathbb{G}}$. On a perfectly sound CRS (where $\vec{f_3} = \vec{f_1}^{\xi_1} \odot \vec{f_2}^{\xi_2}$ and $\vec{\varphi} = \vec{f_3} \odot (1,1,g)$), commitments to exponents are not fully extractable since the trapdoor (α_1, α_2) only allows recovering g^x from $\vec{C} = \vec{\varphi}^x \odot \vec{f_1}^r \odot \vec{f_2}^s$. In order to commit to $x \in \mathbb{Z}_p$, we will sometimes commit to the group element g^x. The result of this process will be denoted by $\vec{C} = \mathsf{GSCom}'(x, \mathit{open}_x) = \mathsf{GSCom}(g^x, \mathit{open}_x)$ with $\mathit{open}_x = (r,s,t)$.

To prove that committed variables satisfy a set of relations, the Groth-Sahai techniques require one commitment per variable and one proof element (made of a constant number of group elements) per relation. Such proofs are available for pairing-product relations, which are of the type

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^{n} \cdot \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = t_T,$$

for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathbb{G}$, $a_{ij} \in \mathbb{G}$, for $i, j \in [1, n]$. Efficient proofs also exist for multi-exponentiation equations

$$\prod_{i=1}^{m} \mathcal{A}_i^{y_i} \cdot \prod_{j=1}^{n} \mathcal{X}_j^{b_j} \cdot \prod_{i=1}^{m} \cdot \prod_{j=1}^{n} \mathcal{X}_j^{y_i \gamma_{ij}} = T,$$

for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$, $y_1, \ldots, y_m \in \mathbb{Z}_p$ and constants $T, \mathcal{A}_1, \ldots, \mathcal{A}_m \in \mathbb{G}$, $b_1, \ldots, b_n \in \mathbb{Z}_p$ and $\gamma_{ij} \in \mathbb{G}$, for $i \in [1, m]$, $j \in [1, n]$.

Multi-exponentiation equations admit zero-knowledge proofs at no additional cost. On a simulated CRS (prepared for the WI setting), the trapdoor (ξ_1, ξ_2) makes it is possible to simulate proofs without knowing witnesses and simulated proofs are perfectly indistinguishable from real proofs. As for pairing-product equations, NIZK proofs are often possible (this is typically the case when the target element t_T has the special form $t_T = \prod_{i=1}^{t} e(S_i, T_i)$, for constants $\{(S_i, T_i)\}_{i=1}^{t}$ and some $t \in \mathbb{N}$) but usually come at some expense.

As far as efficiency goes, quadratic pairing product equations cost 9 elements to prove whereas linear ones (when $a_{ij} = 0$ for all i, j) take 3 group elements. Linear multi-exponentiation equations of the type $\prod_{i=1}^{m} \mathcal{A}_i^{y_i} = T$ demand 2 group elements.

3 A Construction for Inner Product Relations

As noted in [38], many predicates can be expressed in terms of the inner product of two vectors of attributes. In this section, we describe a P-signature scheme for families $(\mathcal{R}_1, \mathcal{R}_2)$ where $\mathcal{R}_1$ encompasses (in)-equality relations and $\mathcal{R}_2$ relates to inner products. Namely, we set $\mathcal{R}_1 = \{R^{\mathsf{EQ}}, R^{\neg\mathsf{EQ}}\}$ and $\mathcal{R}_2 = \{R^{\mathsf{IP}}, R^{\neg\mathsf{IP}}\}$, which are specified as follows. We let $\mathcal{D} = \mathbb{Z}_p$, for some prime p and, for vectors $\vec{m} \in \mathbb{Z}_p^n$, $\vec{X} \in \mathbb{Z}_p^n$, the relations R^{IP} and $R^{\neg\mathsf{IP}}$ are only defined for $i = 0$ in such a way that $R^{\mathsf{IP}}(0, \vec{m}, \vec{X}) = 1$ (resp. $R^{\neg\mathsf{IP}}(0, \vec{m}, \vec{X}) = 1$) if and only if $\vec{m} \cdot \vec{X} = 0$ (resp. $\vec{m} \cdot \vec{X} \neq 0$). As for $\mathcal{R}_1$, we define relations R^{EQ} and $R^{\neg\mathsf{EQ}}$ for $i \in [1,n]$ and so that $R^{\mathsf{EQ}}(i,\vec{m},\vec{X}) = 1$ (resp. $R^{\neg\mathsf{EQ}}(i,\vec{m},\vec{X}) = 1$) if and only if $m_i = x_i$ (resp. $m_i \neq x_i$).

The construction is based on the commitment scheme of section 2.2 and a signature scheme suggested in [24] to sign group elements. The intuition is to sign a commitment to a vector $\vec{m}$ using a signature scheme for group elements such as [24, 28, 1]. Here, a lightweight version of the scheme can be used since, in the proof, the simulator knows the discrete logarithms of the group elements that are signed (hence, there is no need to combine the scheme with a trapdoor commitment to group elements as in [24]). In this simplified version, the signer holds a public key comprising $(\Omega = g^{\omega}, A = g^{\gamma}, u, U_0, U_1 = g^{\beta_1}) \in \mathbb{G}^6$, for private elements $(\omega, \gamma, \beta_1)$. To sign a vector $\vec{m}$, the signer first computes a commitment V to $\vec{m}$, chooses $c \xleftarrow{R} \mathbb{Z}_p$ and computes $\sigma_1 = (g^{\gamma})^{1/(\omega+c)}$, $\sigma_2 = g^c$, $\sigma_3 = u^c$, $\sigma_4 = (U_0 \cdot V^{\beta_1})^c$, $\sigma_5 = V^c$ and also sets σ_6 as part of the signature.

The construction handles inner products using the properties of the commitment scheme recalled in section 2.2. More precisely, we use the property that this scheme allows the committer to generate a short non-interactive argument allowing to convince the verifier that the committed vector $\vec{m}$ is orthogonal to a public vector $\vec{X} = (x_1, \ldots, x_n)$ without revealing anything else. Concretely, given a commitment $C = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$ to $\vec{m} = (m_1, \ldots, m_n)$, for each $i \in [1,n]$, we know that the witness $W_i = g_i^r \cdot \prod_{j=1, j\neq i}^{n} g_{n+1-j+i}^{m_j}$ satisfies

$$e(g_i, C) = e(g_1, g_n)^{m_i} \cdot e(g, W_i), \tag{1}$$

For each i, if we raise both members of (1) to the power x_i and multiply the resulting n equations altogether, we find

$$e\big(\prod_{i=1}^{n} g_i^{x_i}, C\big) = e(g_1, g_n)^{\vec{m}\cdot\vec{X}} \cdot e\big(g, \prod_{i=1}^{n} W_i^{x_i}\big), \tag{2}$$

which implies $e\big(\prod_{i=1}^{n} g_i^{x_i}, C\big) = e\big(g, \prod_{i=1}^{n} W_i^{x_i}\big)$ whenever $\vec{m}\cdot\vec{X} = 0$. As it turns out, a single group element $W = \prod_{i=1}^{n} W_i^{x_i}$ suffices to convince the verifier that $\vec{m}\cdot\vec{X} = 0$. It can be showed (as in the proof in appendix B) that, after the commitment phase, if the committer is able to produce a witness W satisfying $e\big(\prod_{i=1}^{n} g_i^{x_i}, C\big) = e\big(g, W\big)$ and subsequently open the commitment C to a vector $\vec{m}$ such that $\vec{m}\cdot\vec{X} \neq 0$, the n-DHE assumption can be broken.

Likewise, the committer can also convince the verifier that $\vec{m}\cdot\vec{X} \neq 0$ by proving knowledge of group elements $W = \prod_{i=1}^{n} W_i^{x_i}$, $W_1 = g_1^{\vec{m}\cdot\vec{X}} \in \mathbb{G}$ such that

$$e\big(\prod_{i=1}^{n} g_i^{x_i}, C\big) = e(W_1, g_n) \cdot e(g, W). \tag{3}$$

To convince the verifier that $W_1 \neq 1_{\mathbb{G}}$, the prover demonstrates knowledge of another group element $W_0 = g^{1/\vec{m}\cdot\vec{X}}$ for which $e(W_0, W_1) = e(g, g_1)$. We would like to argue that a malicious committer cannot open a commitment C to a vector $\vec{m}$ such that $\vec{m}\cdot\vec{X} = 0$ and also produce $(W, W_0, W_1) \in \mathbb{G}$ such that the equalities $e(W_0, W_1) = e(g, g_1)$ and (3) are both satisfied. Unfortunately, this is not true since a cheating prover can commit to $\vec{m} = \vec{0}$ (which is orthogonal to everything). Since the commitment $C = g^r$ and the value $W = \prod_{i=1}^{n} g_i^{r\cdot x_i}$ satisfy $e(\prod_{i=1}^{n} g_i^{x_i}, C) = e(g, W)$, the prover can fool the verifier by revealing $(W_0, W_1, W') = \big(g_1^{1/\mu}, g^{\mu}, W/g_n^{\mu}\big)$, with $\mu \xleftarrow{R} \mathbb{Z}_p$, which satisfies $e(W_0, W_1) = e(g, g_1)$ and $e(\prod_{i=1}^{n} g_i^{x_i}, C) = e(W_1, g_n) \cdot e(g, W')$.

To address this problem, we require the prover to additionally reveal $(W_2, W_3) = (g^{\vec{m}\cdot\vec{X}}, g_{2n}^{\vec{m}\cdot\vec{X}})$

when stating that $\vec{m} \cdot \vec{X} \neq 0$. The extra checks $e(W_1, g) = e(g_1, W_2)$ and $e(W_1, g_{2n}) = e(g_1, W_3)$ then suffice to convince the verifier. Under the n-FlexDHE assumption, we can show (as detailed in appendix B) that the prover cannot generate witnesses (W_0, W_1, W_2, W_3, W) and subsequently open the commitment to a vector $\vec{m}$ that contradicts the assertion.

Setup(λ)**:** chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ with a generator $g \xleftarrow{R} \mathbb{G}$. It generates a perfectly sound Groth-Sahai CRS $\mathbf{f} = (\vec{f_1}, \vec{f_2}, \vec{f_3})$. Public parameters consist of $\mathsf{params} := \big((\mathbb{G}, \mathbb{G}_T),\ g,\ \mathbf{f}\big)$.

SigSetup(λ, n)**:** picks $\gamma, \omega, \alpha, \beta_1 \xleftarrow{R} \mathbb{Z}_p$, $u, U_0 \xleftarrow{R} \mathbb{G}$ and computes $\Omega = g^{\omega}$, $A = g^{\gamma}$, $U_1 = g^{\beta_1}$ as well as $g_i = g^{(\alpha^i)}$ for each $i \in [1, n] \cup [n + 2, 2n]$. The private key is $\mathsf{sk} = (\gamma, \omega, \beta_1)$ and the public key is defined to be $\mathsf{pk} = \big(u,\ \Omega = g^{\omega},\ A = g^{\gamma},\ U_0,\ U_1,\ \{g_i\}_{i \in [1,2n] \setminus \{n+1\}}\big)$.

Sign$(\mathsf{sk}, \vec{m})$**:** to sign $\vec{m} = (m_1, \ldots, m_n)$, conduct the following steps.

1. Pick $r \xleftarrow{R} \mathbb{Z}_p$ and compute $V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j} = g_n^{m_1} \cdots g_1^{m_n} \cdot g^r$.
2. Choose $c \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\sigma_1 = g^{\gamma/(\omega+c)}, \qquad \sigma_2 = g^c, \qquad \sigma_3 = u^c, \qquad \sigma_4 = (U_0 \cdot V^{\beta_1})^c, \qquad \sigma_5 = V^c, \qquad \sigma_6 = V$$

and output $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$.

Verify$(\mathsf{pk}, \vec{m}, \sigma)$**:** parse σ as $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ and $\vec{m}$ as $(m_1, \ldots, m_n)$.

1. Return 0 if the following equalities do not hold

$$e(A, g) = e(\sigma_1, \Omega \cdot \sigma_2), \qquad e(u, \sigma_2) = e(\sigma_3, g), \qquad e(g, \sigma_4) = e(U_0, \sigma_2) \cdot e(U_1, \sigma_5), \tag{4}$$
$$e(g, \sigma_5) = e(\sigma_6, \sigma_2). \tag{5}$$

2. Return 1 if $\sigma_6 = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$ and 0 otherwise.

Witness-Gen$(\mathsf{pk}, R, i, \vec{m}, \vec{X}, \sigma)$**:** parse σ as $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$. Parse $\vec{m}$ and $\vec{X}$ as $(m_1, \ldots, m_n)$ and $(x_1, \ldots, x_n)$, respectively and return $\perp$ if $\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) = 0$.

a. If $R = R^{\mathsf{EQ}}$ (and $i \in [1, n]$), return $\perp$ if $m_i \neq x_i$. Otherwise, compute and output the witness $W = g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j}$.

b. If $R = R^{\neg\mathsf{EQ}}$ (and $i \in [1, n]$), return $\perp$ if $m_i = x_i$. Otherwise, compute $W_0 = g^{1/(m_i - x_i)}$, $W_1 = g_1^{m_i - x_i}$, $W_2 = g^{m_i - x_i}$, $W_3 = g_{2n}^{m_i - x_i}$ and $W_4 = g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j}$. Return the witness $W = (W_0, W_1, W_2, W_3, W_4)$.

c. If $R = R^{\mathsf{IP}}$ (and $i = 0$), return $\perp$ if $\vec{m} \cdot \vec{X} \neq 0$. Otherwise, compute $W_i = g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j}$ for $i = 1$ to n. Then, compute and output the witness $W = \prod_{i=1}^{n} W_i^{x_i}$.

d. If $R = R^{\neg\mathsf{IP}}$ (and $i = 0$), return $\perp$ if $\vec{m} \cdot \vec{X} = 0$. Otherwise, compute $W_0 = g^{1/(\vec{m} \cdot \vec{X})}$, $W_1 = g_1^{\vec{m} \cdot \vec{X}}$, $W_2 = g^{\vec{m} \cdot \vec{X}}$, $W_3 = g_{2n}^{\vec{m} \cdot \vec{X}}$. For $i = 1$ to n, compute $W_{4,i} = g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j}$ and finally set $W_4 = \prod_{i=1}^{n} W_{4,i}^{x_i}$. Return the witness $W = (W_0, W_1, W_2, W_3, W_4)$.

Witness-Verify$(\mathsf{pk}, R, i, \vec{X}, W, \sigma)$**:** parse σ as $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ and $\vec{X}$ as $(x_1, \ldots, x_n)$. Return 0 if equations (4)-(5) are not satisfied. Otherwise, two cases are distinguished.

a. If $R = R^{\mathsf{EQ}}$ (and $i \in [1, n]$), return 1 iff $e(g_i, \sigma_6) = e(g_1, g_n)^{x_i} \cdot e(g, W)$.

b. If $R = R^{\neg\mathsf{EQ}}$ (and $i \in [1, n]$), parse W as $(W_0, W_1, W_2, W_3, W_4) \in \mathbb{G}^5$ and return $\perp$ if it does not parse properly. Return 1 iff $e\big(g_i, \sigma_6 \cdot g_{n+1-i}^{-x_i}\big) = e(W_1, g_n) \cdot e(g, W_4)$ and[2]

$$e(W_0, W_1) = e(g, g_1), \qquad e(W_1, g) = e(g_1, W_2), \qquad e(W_1, g_{2n}) = e(g_1, W_3). \tag{6}$$

c. If $R = R^{\mathsf{IP}}$ (and $i = 0$), parse W as $W \in \mathbb{G}$ and return 1 iff $e(g, W) = e\big(\prod_{i=1}^{n} g_i^{x_i}, \sigma_6\big)$.

d. If $R = R^{\neg\mathsf{IP}}$ (and $i = 0$), parse W as $(W_0, W_1, W_2, W_3, W_4) \in \mathbb{G}^5$ and return $\perp$ if it does not parse properly. Return 1 iff $e\big(\prod_{i=1}^{n} g_i^{x_i}, \sigma_6\big) = e(W_1, g_n) \cdot e(g, W_4)$ and

$$e(W_0, W_1) = e(g, g_1), \qquad e(W_1, g) = e(g_1, W_2), \qquad e(W_1, g_{2n}) = e(g_1, W_3). \tag{7}$$

The correctness of algorithms Sign and Verify is almost straightforward and that of Witness-Gen and Witness-Verify follows from the properties of the commitment scheme in section 2.2.

P-Signature Protocols. To obtain a complete P-signature, the scheme is augmented with algorithms $\mathsf{SigProve}_i$, for $i \in \{1, 2\}$, and EqComProve.

$\mathsf{SigProve}_1(\mathsf{pk}, R, i, \varUpsilon = \{i\}, \sigma, \vec{m}, \vec{X})$: parse σ as $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$, $\vec{m}$ and $\vec{X}$ as $(m_1, \ldots, m_n)$ and $(x_1, \ldots, x_n)$, respectively. Then, compute $\{\vec{C}_{x_{t,j}} = \mathsf{GSCom}(X_{t,j}, open_{x_{t,j}})\}_{t \in \varUpsilon, j \in \{1,2,3\}}$ as commitments to the variables $\{(X_{t,1}, X_{t,2}, X_{t,3}) = (g_1^{x_t}, g^{x_t}, g_{2n}^{x_t})\}_{t \in \varUpsilon}$. For $j = 1$ to 6, compute $\vec{C}_{\sigma_j} = \mathsf{GSCom}(\sigma_j, open_{\sigma_j})$ and generate a NIZK proof that committed variables $\{\sigma_j\}_{j=1}^{6}$ satisfy (4)-(5). This requires to introduce auxiliary variables $\sigma_7 \in \mathbb{G}$, $\theta_1 \in \mathbb{Z}_p$ with their own commitments $\vec{C}_{\sigma_7} = \mathsf{GSCom}(\sigma_7, open_{\sigma_7})$, $\vec{C}_{\theta_1} = \mathsf{GSCom}(\theta_1, open_{\theta_1})$ and to prove that

$$e(\sigma_7, g) = e(\sigma_1, \varOmega \cdot \sigma_2), \qquad e(u, \sigma_2) = e(\sigma_3, g), \tag{8}$$
$$e(g, \sigma_4) = e(U_0, \sigma_2) \cdot e(U_1, \sigma_5), \qquad e(g, \sigma_5) = e(\sigma_6, \sigma_2), \tag{9}$$
$$\theta_1 = 1, \qquad e(A/\sigma_7, g^{\theta_1}) = 1_{\mathbb{G}_T} \tag{10}$$

Let π_σ be the proof for relations (8)-(10). Then, the algorithm considers two cases.

- If $R = R^{\mathsf{EQ}}$, let $\vec{C}_W = \mathsf{GSCom}(W, open_W)$, with $W = \mathsf{Witness\text{-}Gen}(\mathsf{pk}, R^{\mathsf{EQ}}, i, \vec{m}, \vec{X}, \sigma)$. Generate proofs π_{x_i}, $\{\pi_{X_{t,j}}\}_{t \in \varUpsilon, j=1,2}$ that committed variables σ_6, W and $X_{i,1}$ satisfy

$$e(g_i, \sigma_6) = e(X_{i,1}, g_n) \cdot e(g, W), \tag{11}$$
$$e(X_{i,2}, g_1) = e(X_{i,1}, g), \qquad e(X_{i,2}, g_{2n}) = e(X_{i,3}, g). \tag{12}$$

  The final proof is $\pi = \big(\{\vec{C}_{x_{t,j}}\}_{t \in \varUpsilon, j \in \{1,2,3\}}, \{\vec{C}_{\sigma_j}\}_{j=1}^{7}, \vec{C}_W, \vec{C}_{\theta_1}, \pi_\sigma, \pi_{x_i}, \{\pi_{X_{t,j}}\}_{t \in \varUpsilon, j=1,2}\big)$.

- If $R = R^{\neg\mathsf{EQ}}$, generate commitments $\{C_{W_j}\}_{j=0}^{4}$ to the variables $(W_0, W_1, W_2, W_3, W_4) \leftarrow \mathsf{Witness\text{-}Gen}(\mathsf{pk}, R^{\neg\mathsf{EQ}}, i, \vec{m}, \vec{X}, \sigma)$. Generate proofs π_{x_i}, π_W for relations (13) and (14)

$$e(g_i, \sigma_6) \cdot e(X_{i,1}, g_n)^{-1} = e(W_1, g_n) \cdot e(g, W_4) \tag{13}$$
$$e(W_0, W_1) = e(g, g_1) \qquad e(W_1, g) = e(g_1, W_2) \qquad e(W_1, g_{2n}) = e(g_1, W_3), \tag{14}$$

  and proofs $\{\pi_{X_{t,j}}\}_{t \in \varUpsilon, j=1,2}$ that $\{(X_{t,1}, X_{t,2}, X_{t,3})\}_{t \in \varUpsilon}$ satisfy (12). The final proof consists of $\pi = \big(\{\vec{C}_{x_{t,j}}\}_{t \in \varUpsilon, j \in \{1,2,3\}} \{\vec{C}_{\sigma_j}\}_{j=1}^{7}, \{\vec{C}_{W_i}\}_{j=0}^{4}, \vec{C}_{\theta_1}, \pi_\sigma, \pi_{x_i}, \pi_W, \{\pi_{X_{t,j}}\}_{t \in \varUpsilon, j=1,2}\big)$.

[2] Looking ahead, W_0 will be useful to convince the verifier (via the first relation of (7)) that $W_1 \neq 1_{\mathbb{G}}$ when (W_1, W_2, W_3, W_4) will appear in committed form within Groth-Sahai proofs produced by $\mathsf{SigProve}_2$. Although W_0 is not strictly necessary in Witness-Verify in the cases $R = R^{\neg\mathsf{IP}}$ and $R = R^{\neg\mathsf{EQ}}$ (since the algorithm can directly check that $W_1 \neq 1_{\mathbb{G}}$), we included it among the outputs of Witness-Gen for ease of explanation.

$\mathsf{SigProve}_2(\mathsf{pk}, R, i, \sigma, \vec{m}, \vec{X})$: parse σ and $\vec{m}$ as previously and $\vec{X}$ as $(x_1, \ldots, x_n)$. For $i = 1$ to 6, compute $\vec{C}_{\sigma_i} = \mathsf{GSCom}(\sigma_i, \mathit{open}_{\sigma_i})$. Using extra variables $\sigma_7 \in \mathbb{G}$, $\theta_1 \in \mathbb{Z}_p$ and their commitments $\vec{C}_{\sigma_7} = \mathsf{GSCom}(\sigma_7, \mathit{open}_{\sigma_7})$, $\vec{C}_{\theta_1} = \mathsf{GSCom}(\theta_1, \mathit{open}_{\theta_1})$, generate a NIZK proof that $\{\sigma_i\}_{i=1}^{6}$ satisfy (4)-(5). We call π_σ the proof for (8)-(10). Then, consider the two following cases.

- If $R = R^{\mathsf{IP}}$, set $\vec{C}_W = \mathsf{GSCom}(W, \mathit{open}_W)$, where $W = \mathsf{Witness\text{-}Gen}(\mathsf{pk}, R^{\mathsf{IP}}, 0, \vec{m}, \vec{X}, \sigma)$. Then, generate a proof $\pi_{\vec{X}}$ that W and σ_6 satisfy

$$e\Big(\prod_{j=1}^{n} g_j^{x_j}, \sigma_6\Big) = e(g, W). \tag{15}$$

  The NIZK proof is $\pi = \big(\{\vec{C}_{\sigma_j}\}_{j=1}^{7}, \vec{C}_W, \vec{C}_{\theta_1}, \pi_\sigma, \pi_{\vec{X}}\big)$.

- If $R = R^{\neg\mathsf{IP}}$, define the auxiliary variable $\Theta = g \in \mathbb{G}$ and generate $\vec{C}_\Theta = \mathsf{GSCom}(\Theta, \mathit{open}_\Theta)$, $\{\vec{C}_{W_j} = \mathsf{GSCom}(W_j, \mathit{open}_{W_j})\}_{j=0}^{4}$, where $\{W_j\}_{j=0}^{4} \leftarrow \mathsf{Witness\text{-}Gen}(\mathsf{pk}, R^{\neg\mathsf{IP}}, 0, \vec{m}, \vec{X}, \sigma)$. Then, generate a proof $\pi_{\vec{X}}^{\mathsf{NOT}}$ that Θ and $\{W_i\}_{i=0}^{4}$ satisfy

$$e(W_0, W_1) = e(\Theta, g_1), \qquad e(W_1, g) = e(g_1, W_2), \tag{16}$$
$$e(W_1, g_{2n}) = e(g_1, W_3), \qquad e(\Theta/g, g^{\theta_1}) = 1_{\mathbb{G}_T}. \tag{17}$$
$$e\Big(\prod_{j=1}^{n} g_j^{x_j}, \sigma_6\Big) = e(g, W_4) \cdot e(W_1, g_n). \tag{18}$$

  The NIZK proof consists of $\pi = \big(\{\vec{C}_{\sigma_j}\}_{j=1}^{7}, \{\vec{C}_{W_j}\}_{j=0}^{4}, \vec{C}_{\theta_1}, \pi_\sigma, \pi_{\vec{X}}^{\mathsf{NOT}}\big)$.

- If $R = R^{\mathsf{EQ}}$ (and $i \in [1, n]$), the algorithm generates $\vec{C}_W = \mathsf{GSCom}(W, \mathit{open}_W)$ where $W \leftarrow \mathsf{Witness\text{-}Gen}(\mathsf{pk}, R^{\mathsf{EQ}}, i, \vec{m}, \vec{X}, \sigma)$, introduces a commitment $\vec{C}_{X_i} = \mathsf{GSCom}(X_i, \mathit{open}_{X_i})$ to the auxiliary variable $X_i = g_1^{x_i}$ and compute proofs π_W and π_{x_i} that

$$e(g_i, \sigma_6) = e(X_i, g_n) \cdot e(g, W) \qquad e(X_i/g_1^{x_i}, g^{\theta_1}) = 1_{\mathbb{G}_T} \tag{19}$$

  The proof is $\pi = \big(\{\vec{C}_{\sigma_j}\}_{j=1}^{7}, \vec{C}_W, \vec{C}_{X_i}, \vec{C}_{\theta_1}, \pi_\sigma, \pi_W, \pi_{x_i}\big)$.

- If $R = R^{\neg\mathsf{EQ}}$ (and thus $i \in [1, n]$), compute $\{\vec{C}_{W_j} = \mathsf{GSCom}(W_j, \mathit{open}_{W_j})\}_{j=0}^{4}$ where $\{W_j\}_{j=0}^{4} \leftarrow \mathsf{Witness\text{-}Gen}(\mathsf{pk}, R^{\neg\mathsf{EQ}}, i, \vec{m}, \vec{X}, \sigma)$, introduce $\vec{C}_{X_i} = \mathsf{GSCom}(X_i, \mathit{open}_{X_i})$ for the auxiliary variable $X_i = g_1^{x_i}$ and generate proofs $(\pi_{X_i,W}, \{\pi_{W_j}\}_{j=1}^{3}, \pi_{X_i}, \pi_\Theta)$ for

$$e(g_i, \sigma_6) \cdot e(X_i, g_n)^{-1} = e(W_1, g_n) \cdot e(g, W), \tag{20}$$
$$e(W_0, W_1) = e(\Theta, g_1), \qquad e(W_1, g) = e(g_1, W_2), \qquad e(W_1, g_{2n}) = e(g_1, W_3), \tag{21}$$
$$e(X_i/g_1^{x_i}, g^{\theta_1}) = e(\Theta/g, g^{\theta_1}) = 1_{\mathbb{G}_T} \tag{22}$$

  The proof is $\pi = \big(\{\vec{C}_{\sigma_j}\}_{j=1}^{7}, \{\vec{C}_{W_j}\}_{j=0}^{4}, \vec{C}_{X_i}, \vec{C}_{\theta_1}, \pi_\sigma, \pi_{X_i,W}, \{\pi_{W_j}\}_{j=1}^{3}, \pi_{X_i}, \pi_\Theta\big)$.

$\mathsf{SigIssue}\big(\mathsf{sk}, V', (m_{n_1+1}, \ldots, m_n)\big) \leftrightarrows \mathsf{SigObtain}(\mathsf{pk}, \vec{m}_{|n_1} = (m_1, \ldots, m_{n_1}), \mathit{open}_{\vec{m}_{|n_1}})$: the user $\mathcal{U}$ and the issuer interact with each other in the following way.

1. $\mathcal{U}$ commits to $\vec{m}_{|n_1} = (m_1, \ldots, m_{n_1})$ and computes $V' = g^{r'} \cdot \prod_{j=1}^{n_1} g_{n+1-j}^{m_j}$, where $r' \stackrel{R}{\leftarrow} \mathbb{Z}_p$, retains $\mathit{open}_{\vec{m}_{|n_1}} = (m_1, \ldots, m_{n_1}, r')$ and provides the issuer with an interactive WI proof of knowledge of $(m_1, \ldots, m_{n_1}, r')$ such that $V' = g^{r'} \cdot \prod_{j=1}^{n_1} g_{n+1-j}^{m_j}$.

2. The issuer sets $V = V' \cdot \prod_{j=n_1+1}^{n} g_{n+1-j}^{m_j}$. Then, it randomly chooses $c, r'' \overset{R}{\leftarrow} \mathbb{Z}_p$, computes $\sigma_1 = g^{\gamma/(\omega+c)}$, $\sigma_2 = g^c$, $\sigma_3 = u^c$ and

$$\sigma_4 = \big(U_0 \cdot (V \cdot g^{r''})^{\beta_1}\big)^c, \qquad \sigma_5 = (V \cdot g^{r''})^c, \qquad \sigma_6 = V \cdot g^{r''}$$

and returns $\tilde{\sigma} = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6)$ and r''.

3. $\mathcal{U}$ outputs $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$, where $r = r' + r''$.

The algorithm $\mathsf{EqComProve}$ for generating proofs that two commitments open to the same value is standard: if $\vec{C_X} = \mathsf{GSCom}(X, open_X)$ and $\vec{C_Y} = \mathsf{GSCom}(Y, open_Y)$ are Groth-Sahai commitments to $X = Y \in \mathbb{G}$, the NIZK proof can be a proof that $\vec{C_X} \odot \vec{C_Y}^{-1}$ is a commitment that opens to $1_{\mathbb{G}}$. If we write $\vec{f_1} = (f_1, 1, g)$, $\vec{f_2} = (1, f_2, g)$ and $\vec{f_3} = (f_{31}, f_{32}, f_{33})$, this amounts to proving the existence of $(\rho_1, \rho_2, \rho_3) \in \mathbb{Z}_p^3$ such that $\vec{C_X} \odot \vec{C_Y}^{-1} = (f_1^{\rho_1} \cdot f_{31}^{\rho_3}, f_2^{\rho_2} \cdot f_{32}^{\rho_3}, g^{\rho_1+\rho_2} \cdot f_{33}^{\rho_3})$. On a simulated CRS, this relation can always be proved in NIZK since it is a linear multi-exponentiation equation.

Efficiency. From an efficiency standpoint, the outputs of $\mathsf{SigProve}_1$ consist of 80 elements of $\mathbb{G}$ for R^{EQ} and 101 group elements for $R^{\neg\mathsf{EQ}}$. Each proof produced by $\mathsf{SigProve}_2$ requires less than 80 group elements for relations R^{EQ} and R^{IP} and at most 107 elements in the case of $R^{\neg\mathsf{EQ}}$ and $R^{\neg\mathsf{IP}}$.

When these proofs are combined to prove the ownership of a credential, they result in non-interactive proofs demanding about 2 kB at the 80-bit security level. A detailed efficiency analysis is provided in appendix E.3.

We leave it as an interesting open problem to eliminate the dependency on n in the public key size (as was done in [15]) without using interaction or random oracles.

Security. The security of the scheme relies on the assumptions described at the beginning of section 2. The proofs of the following theorems are available in appendices B, C and D, respectively.

Theorem 1. *If the HSDH, FlexDH and n-FlexDHE assumptions hold in $\mathbb{G}$, the above block-wise P-signature scheme is $(F, \mathcal{R}_1, \mathcal{R}_2)$-unforgeable w.r.t. the injective function $F(m) = (g_1^m, g^m, g_{2n}^m)$ and the relations families $\mathcal{R}_1 = \{R^{\mathsf{EQ}}, R^{\neg\mathsf{EQ}}\}$, $\mathcal{R}_2 = \{R^{\mathsf{IP}}, R^{\neg\mathsf{IP}}\}$.*

Theorem 2. *The block-wise P-signature provides signer and user privacy if the underlying WI proof of knowledge is secure.*

Theorem 3. *The block-wise P-signature is zero-knowledge if the DLIN assumption holds in $\mathbb{G}$.*

4 Non-Interactive Anonymous Credentials with Efficient Attributes

In appendix E, we provide the complete details about how block-wise P-signatures for these relation families can be generically turned into non-interactive anonymous credentials with efficient attributes. Proper security definitions for these are given in appendix E.1 and we prove the security of the generic construction in the same way as in [3] in appendix E.2.

In a nutshell, the construction appeals to $\mathsf{SigProve}_1$ to prove that the first component of the user's certified vector $\vec{m}$ is his private key sk_U which the user's pseudonym is a commitment of. Then, $\mathsf{SigProve}_2$ is used to convince the verifier that the certified vector $\vec{X}$ satisfies $\vec{m} \cdot \vec{X} = 0$. As described in appendix E.2, the construction is presented without optimizations for the sake of generality. Its optimized variant provides proofs of about 2 kB.

Appendix E.4 summarizes the predicates that can be expressed using inner product relations and suitable attribute encodings (already used in [38]). For example, when $\vec{m}$ contains the coefficients a polynomial whose roots are the user's attributes, the inclusion (or the non-inclusion) of some attribute $\omega \in \mathbb{Z}_p$ can be selectively demonstrated by setting the coordinates of $\vec{X}$ as $(1, \omega, \ldots, \omega^{n-1})$. A similar technique can be used to prove that some certified attribute ω (this time encoded as a sub-vector $(1, \omega, \ldots, \omega^{n-1})$ of $\vec{m}$) lies in a public list (or not) by proving its orthogonality to some $\vec{X}$ that contains the coefficients of a polynomial.

Using more complex attribute encodings, inner products can also handle disjunctions of a small (e.g., logarithmic in λ) number of atomic conditions. If we assume only two rounds of interaction, conjunctions can also be dealt with: the verifier just has to send a short random challenge in $\mathbb{Z}_p$ which is used to randomize the vector $\vec{X}$ in such a way that the condition $\vec{m} \cdot \vec{X} = 0$ guarantees the validity of assertions $(m_1 = x_1) \wedge \ldots \wedge (m_n = x_n)$ with overwhelming probability. Although the need for interaction seems at odds with the original motivation of P-signatures, we still gain something since only two rounds are necessary.

Finally, as already noted in [38], inner products also provide a method to prove exact threshold statements about sets of binary attributes. For example, if $\vec{m}$ and $\vec{X}$ encode two sets of binary attributes (such as "gender", "graduated", etc.) X and S, the prover can convince the verifier that $|S \cap X| = t$. In addition, by combining the same technique with set membership proofs [13], statements about inexact thresholds $|S \cap X| \leq t$ can also be proved as detailed in appendix E.4.

References

1. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo. Structure-Preserving Signatures and Commitments to Group Elements. In *Crypto'10*, LNCS 6223, pp. 209–236, 2010.
2. N. Akagi, Y. Manabe, T. Okamoto. An Efficient Anonymous Credential System. In *Financial Cryptography (FC'08)*, LNCS 5143, pp. 272–286, 2008.
3. M. Belenkiy, M. Chase, M. Kohlweiss and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC'08*, LNCS 4948, pages 356–374, 2008.
4. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya and H. Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In *Crypto'09*, LNCS 5677, pp. 108–125, 2009
5. M. Belenkiy, M. Chase, M. Kohlweiss and A. Lysyanskaya. Compact E-Cash and Simulatable VRFs Revisited. In *Pairing'09*, LNCS 5671, pp. 114–131, 2009.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, pp. 62–73, 1993.
7. O. Blazy, G. Fuchsbauer, M. Izabachène, A. Jambert, H. Sibert, D. Vergnaud. Batch Groth-Sahai. In *Applied Cryptography and Network Security (ACNS'10)*, LNCS 6123, pp. 218–235, 2010.
8. D. Boneh and X. Boyen. Short signatures without random oracles. In *Eurocrypt'04*, LNCS 3027, pages 56–73, 2004.
9. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. In *Crypto'04*, *LNCS* 3152, pp. 41–55, 2004.
10. D. Boneh, C. Gentry and B. Waters. Collusion-Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Crypto'05*, *LNCS* 3621, pp. 258–275, 2005.
11. X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC'07* , LNCS 4450, pp. 1–15, 2007.
12. S. Brands. Rethinking Public Key Infrastructure and Digital Certificates – Building in Privacy. PhD Thesis, Eindhoven Inst. of Tech., 1999.
13. J. Camenisch, R. Chaabouni and a. shelat. Efficient Protocols for Set Membership and Range Proofs. In *Asiacrypt'08*, LNCS 5330, pp. 234–252, 2008.
14. J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In *Eurocrypt'09*, LNCS 5479, pp. 351–368, 2009.
15. J. Camenisch and T. Groß. Efficient Attributes for Anonymous Credentials. In *ACM-CCS'08*, pp. 345–356, ACM Press, 2008. Extended version available from `http://eprint.iacr.org/2010/496`.
16. J. Camenisch, S. Hohenberger and A. Lysyanskaya. Compact E-Cash. In *Eurocrypt'05*, LNCS 3494, pp. 302–321, 2005.

17. J. Camenisch, M. Kohlweiss and C. Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *PKC'09*, *LNCS* 5443, pp. 481–500, 2009.
18. J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Eurocrypt'01*, *LNCS* 2045, pp. 93–118, 2001.
19. J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN'02*, *LNCS* 2576, pp. 268–289, 2001.
20. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Crypto'04*, *LNCS* 3152, pp. 56–72, 2004.
21. J. Camenisch, E. Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *ACM-CCS'02*, pp. 21–30, ACM Press, 2002.
22. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM, 28(10), pp. 1030–1044, 1985.
23. R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. In *STOC'98*, pp. 209–218, ACM Press, 1998.
24. J. Cathalo, B. Libert and M. Yung. Group Encryption: Non-Interactive Realization in the Standard Model. In *Asiacrypt'09*, *LNCS* 5912, pp. 179–196, 2009.
25. D.-W. Cheung, N. Mamoulis, W.-K. Wong, S.-M. Yiu and Y. Zhang. Anonymous Fuzzy Identity-based Encryption for Similarity Search. In *ISAAC 2010*, *LNCS* 6506, pp. 61–72, 2010.
26. I. Damgård. Payment Systems and Credential Mechanisms with Provable Security Against Abuse by Individuals. In *Crypto'88*, *LNCS* 403, pp. 328–335, 1988.
27. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto'86*, LNCS 263, pp. 186–194, 1986.
28. G. Fuchsbauer. Automorphic Signatures in Bilinear Groups and an Application to Round-Optimal Blind Signatures. Cryptology ePrint Archive: Report 2009/320, 2009.
29. G. Fuchsbauer. Commuting Signatures and Verifiable Encryption and an Application to Non-Interactively Delegatable Credentials. In *Eurocrypt'11*, LNCS 6632, pp. 224–245, 2011.
30. E. Fujisaki, T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *Crypto'97*, *LNCS* 1294, pp. 16–30, 1997.
31. S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2), pp. 281–308, 1988.
32. O. Goldreich, S. Micali, A. Widgerson. Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design. In *FOCS'86*, pp. 174–187, 1986.
33. S. Goldwasser and Y. Tauman-Kalai. On the (In)security of the Fiat-Shamir Paradigm In *FOCS'03*, pages 102–115, 2003.
34. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt'08*, LNCS 4965, pp. 415–432, 2008.
35. M. Jakobsson, K. Sako, R. Impagliazzo. Designated Verifier Proofs and Their Applications. In *Eurocrypt'96*, LNCS 1070, pp. 143–154, 1996.
36. J. Katz. Efficient and Non-malleable Proofs of Plaintext Knowledge and Applications. In *Eurocrypt'03*, LNCS 2656, pp. 211–228, 2003.
37. S. Kunz-Jacques and D. Pointcheval. About the security of MTI/C0 and MQV. In *SCN'06*, LNCS 4116, pp. 156–172, 2006.
38. J. Katz, A. Sahai and B. Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Eurocrypt'08*, *LNCS* 4965, pp. 146-162, 2008.
39. B. Libert and M. Yung. Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs. In *TCC'10*, LNCS 5978, pp. 499–517, 2010.
40. A. Lysyanskaya, R. Rivest, A. Sahai, S. Wolf. Pseudonym Systems. In *Selected Areas in Cryptography (SAC'99)*, LNCS 1758, pp. 184–199, 1999.
41. H.K. Maji, M. Prabhakaran, M. Rosulek. Attribute-based signatures. In *CT-RSA'11*, *LNCS* 6558, pp. 376–392, 2011.
42. T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Crypto'91*, *LNCS* 576, pp. 129–140, 1991.
43. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt'97*, *LNCS* 1233, pp. 256–66, 1997.
44. A. Rial, M. Kohlweiss and B. Preneel. Universally Composable Adaptive Priced Oblivious Transfer. In *Pairing'09*, *LNCS* 5671, pp. 231–247, 2009.
45. S.-F. Shahandashti, R. Safavi-Naini. Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. In *Africacrypt'09*, *LNCS* 5580, pp. 198–216, 2009.

46. J. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM 27, pp. 701717, 1980.

47. A. Yao. How to Generate and Exchange Secrets. In *FOCS'86*, pp. 162–167, 1986.

A Other Security Definitions for Block-wise P-signatures

The notion of unforgeability for P-signatures was formalized in section 2. This section gives proper definitions for the notions of signer and user privacy and the zero-knowledge property.

These definitions are essentially those of Belenkiy *et al.* [3] with minor changes due to need to accommodate to the efficient attribute setting.

Signer Privacy. As formalized in [3], this notion captures that, during its interaction with the honest issuer, an adversary acting as a malicious user should not gain any side information beyond the signature on a vector $\vec{m} = (\vec{m}_{|n_1} | (m_{n_1+1}, \ldots, m_n)) \in \mathcal{D}^n$. More precisely, there must exist an efficient simulator $\mathsf{SimIssue}$ such that no PPT adversary $\mathcal{A}$ can tell whether it is running $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ in interaction with a real issuer or if it is interacting with $\mathsf{SimIssue}$ that only has access to a signing oracle. Formally, a block-wise P-signature ensures *signer privacy* if there is a negligible function $\nu(\lambda)$ such that, for any PPT algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have $\left|\Pr[\mathsf{Exp}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{SimIssue}}(\lambda) = 1]\right| \leq \nu(\lambda)$, where $\mathsf{Exp}_{\mathcal{A}}$ and $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{SimIssue}}$ are experiments defined as follows.

```
$\mathsf{Exp}_{\mathcal{A}}(\lambda)$ :
  $\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SigSetup}(\lambda, n)$
  $(\vec{m}, open_{\vec{m}_{|n_1}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}, \mathsf{sk})$
  $C_{\vec{m}_{|n_1}} \leftarrow \mathrm{VecCom}(\vec{m}_{|n_1}, open_{\vec{m}_{|n_1}})$
  $\mathsf{SigIssue}(\mathsf{sk}, C_{\vec{m}_{|n_1}}, (m_{n_1+1}, \ldots, m_n)) \leftrightarrows \mathcal{A}_2(\mathsf{state}) \Rightarrow b$
  Return b
```

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{SimIssue}}(\lambda)$ :
  $\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SigSetup}(\lambda, n)$
  $(\vec{m}, open_{\vec{m}_{|n_1}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}, \mathsf{sk})$
  $C_{\vec{m}_{|n_1}} \leftarrow \mathrm{VecCom}(\vec{m}_{|n_1}, open_{\vec{m}_{|n_1}})$
  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \vec{m})$
  $\mathsf{SimIssue}(C_{\vec{m}_{|n_1}}, (m_{n_1+1}, \ldots, m_n), \sigma) \leftrightarrows \mathcal{A}_2(\mathsf{state}) \Rightarrow b$
  Return b
```

We note that, as insisted in [3], $\mathsf{SimIssue}$ is allowed to rewind $\mathcal{A}$ if necessary.

User Privacy. User privacy is also defined following [3]. It requires that any malicious signer interacting with an honest user be unable to learn anything about the user's private messages $\vec{m}_{|n_1} \in \mathcal{D}^{n_1}$. As previously, there must exist an efficient simulator $\mathsf{SimObtain}$ – which is allowed to rewind the adversary $\mathcal{A}$ – such that a dishonest signer $\mathcal{A}$ cannot distinguish a conversation with a real user from an interaction with $\mathsf{SimObtain}$. A block-wise P-signature provides *user privacy* if there is a negligible function $\nu(\lambda)$ such that, for any PPT algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that $\left|\Pr[\mathsf{Exp}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{SimObtain}}(\lambda) = 1]\right| \leq \nu(\lambda)$, where $\mathsf{Exp}_{\mathcal{A}}$ and $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{SimObtain}}$ are the experiments defined hereafter. In these two experiments, the adversary is allowed to generate the signer's key pair $(\mathsf{sk}, \mathsf{pk})$ herself but we assume that the user performs a sanity check on the public key pk. As before, $\mathsf{SimObtain}$ is allowed to rewind the adversary.

$\mathsf{Exp}_{\mathcal{A}}(\lambda)$:	$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{SimObtain}}(\lambda)$:
$\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$	$\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{pk}, \vec{m}, open_{\vec{m}_{\lvert n_1}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params})$	$(\mathsf{pk}, \vec{m}, open_{\vec{m}_{\lvert n_1}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params})$
`if` $\mathrm{Valid}(\mathsf{pk}) = 1$	`if` $\mathrm{Valid}(\mathsf{pk}) = 1$
$\quad C_{\vec{m}_{\lvert n_1}} \leftarrow \mathrm{VecCom}(\vec{m}_{\lvert n_1}, open_{\vec{m}_{\lvert n_1}})$	$\quad C_{\vec{m}_{\lvert n_1}} \leftarrow \mathrm{VecCom}(\vec{m}_{\lvert n_1}, open_{\vec{m}_{\lvert n_1}})$
`else output` $\perp$	`else output` $\perp$
$b \Leftarrow \mathcal{A}_2(\mathsf{state}) \leftrightarrows \mathsf{SigObtain}(\mathsf{pk}, \vec{m}_{\lvert n_1}, open_{\vec{m}_{\lvert n_1}})$	$b \Leftarrow \mathcal{A}_2(\mathsf{state}) \leftrightarrows \mathsf{SimObtain}(\mathsf{pk}, C_{\vec{m}_{\lvert n_1}})$
`Return` b	`Return` b

Zero Knowledge. To formalize the zero-knowledge property, we introduce a simulator Sim that implements P-signature algorithms for generating parameters, proving statements involving some relation family $\mathcal{R}$ and proving the equality of commitment openings without using any secret. If the adversary cannot tell whether it is interacting with real algorithms or simulators, the scheme is guaranteed not to leak useful information about secret values.

Formally, a block-wise P-signature protocol is said *zero-knowledge* if there exists a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimSigProve}_1, \mathsf{SimSigProve}_2, \mathsf{SimEqComProve})$, such that for any PPT adversary, simulated parameters params_s are computationally indistinguishable from those produced by Setup, and for all outputs $(\mathsf{params}_s, \tau)$ of $\mathsf{SimSetup}$, it holds that (1) $\mathsf{Com}(\mathsf{params}_s, \cdot)$ is now perfectly hiding; (2) $\mathsf{SimSigProve}_1(\mathsf{params}_s, \tau, \mathsf{pk}, R_1, i, \Upsilon, \vec{X})$ generates identically distributed proofs to those produced by $\mathsf{SigProve}_1(\mathsf{params}_s, \mathsf{pk}, R_1, i, \Upsilon, \sigma, \vec{m}, \vec{X})$ for any $i \in [1, n]$, any signature σ on $\vec{m}$, any relation $R_1 \in \mathcal{R}_1$ and any $\vec{X} \in \mathcal{D}^n$; (3) $\mathsf{SimSigProve}_2(\mathsf{params}_s, \tau, \mathsf{pk}, R, i, \vec{X})$ generates proofs which are indistinguishable from those generated by $\mathsf{SigProve}_2(\mathsf{params}_s, \mathsf{pk}, R, i, \sigma, \vec{m}, \vec{X})$ for any $i \in [0, n]$, for any signature σ on $\vec{m}$, any relation $R \in \mathcal{R}_1 \cup \mathcal{R}_2$ and any $\vec{X} \in \mathcal{D}^n$; (4) $\mathsf{SimEqComProve}(\mathsf{params}_s, \tau, C_x, C_y)$ is indistinguishable to $\mathsf{EqComProve}(\mathsf{params}_s, m, open_x, open_y)$, where $C_x = \mathsf{Com}(x, open_x)$ and $C_y = \mathsf{Com}(y, open_y)$. As before, the previous notions can be defined using a real security game and a simulation. The two games should be indistinguishable even if the adversary is given the simulation trapdoor τ in both games. Using the standard notations of [31], this is formalized by mandating the existence of negligible functions $\nu(.)$ such that the following equalities hold for any relations $R_1 \in \mathcal{R}_1$ and $R \in \mathcal{R}_1 \cup \mathcal{R}_2$:

$$\Big| \Pr\big[\mathsf{params} \leftarrow \mathsf{Setup}(\lambda); b \leftarrow \mathcal{A}(\mathsf{params}) : b = 1\big] \\ - \Pr[(\mathsf{params}_s, \tau) \leftarrow \mathsf{SimSetup}(\lambda); b \leftarrow \mathcal{A}(\mathsf{params}_s) : b = 1]\Big| < \nu(\lambda),$$

$$\Big| \Pr\big[(\mathsf{params}_s, \tau) \leftarrow \mathsf{SimSetup}(\lambda);\ (\mathsf{pk}, i, \sigma, \vec{m}, \vec{X}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}_s, \tau); \\ (\mathbf{C}, \pi) \leftarrow \mathsf{SigProve}_1(\mathsf{params}, \mathsf{pk}, R_1, i, \sigma, \vec{m}, \vec{X}); b \leftarrow \mathcal{A}_2\,(\mathsf{state}, \mathbf{C}, \pi) : b = 1\big] \\ - \Pr\big[(\mathsf{params}_s, \tau) \leftarrow \mathsf{SimSetup}(\lambda); (\mathsf{pk}, i, \sigma, \vec{m}, \vec{X}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}_s, \tau); \\ (\mathbf{C}, \pi) \leftarrow \mathsf{SimSigProve}_1(\mathsf{params}_s, \tau, \mathsf{pk}, R_1, \vec{X}); b \leftarrow \mathcal{A}_2\,(\mathsf{state}, \mathbf{C}, \pi) : b = 1\big]\Big| < \nu(\lambda),$$

$$\Big| \Pr\big[(\mathsf{params}_s, \tau) \leftarrow \mathsf{SimSetup}(\lambda);\ (\mathsf{pk}, i, \sigma, \vec{m}, \vec{X}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}_s, \tau); \\ (C_W, C_\sigma, \pi) \leftarrow \mathsf{SigProve}_2(\mathsf{params}_s, \mathsf{pk}, R, i, \sigma, \vec{m}, \vec{X}); b \leftarrow \mathcal{A}_2(\mathsf{state}, C_W, C_\sigma, \pi) : b = 1\big] \\ - \Pr\big[(\mathsf{params}_s, \tau) \leftarrow \mathsf{SimSetup}(\lambda); (\mathsf{pk}, i, \sigma, \vec{m}, \vec{X}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}_s, \tau); \\ (C_W, C_\sigma, \pi) \leftarrow \mathsf{SimSigProve}_2(\mathsf{params}_s, \tau, \mathsf{pk}, R_1, \vec{X}); b \leftarrow \mathcal{A}_2\,(\mathsf{state}, C_W, C_\sigma, , \pi) : b = 1\big]\Big| < \nu(\lambda),$$

$$\Big| \Pr\big[(\mathsf{params}_s,\tau) \leftarrow \mathsf{SimSetup}(\lambda);\ (m, open_x, open_y, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}_s,\tau);$$
$$(C_x, C_y, \pi) \leftarrow \mathsf{EqComProve}(\mathsf{params}_s, m, open_x, open_y); b \leftarrow \mathcal{A}_2(\mathsf{state}, C_x, C_y, \pi) : b = 1\big]$$
$$- \Pr\big[(\mathsf{params}_s,\tau) \leftarrow \mathsf{SimSetup}(\lambda); (m, open_x, open_y, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}_s,\tau);$$
$$(C_x, C_y, \pi) \leftarrow \mathsf{SimEqComProve}\big(\mathsf{params}_s, \tau, \mathsf{Com}(m, open_x), \mathsf{Com}(m, open_y)\big);$$
$$b \leftarrow \mathcal{A}_2(\mathsf{state}, C_x, C_y, \pi) : b = 1\big]\Big| < \nu(\lambda)$$

B Proof of Theorem 1

The proof considers three kinds of forgeries in the attack game. In the following notations, we denote by $\sigma_j = (\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \sigma_{j,4}, \sigma_{j,5}, \sigma_{j,6}, r_j)$ the output of the j-th signing query. We also denote by $(\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6, r^\star), \mathsf{Pred}^\star, W^\star)$ the forgery that the adversary $\mathcal{A}$ outputs.

- Type I F-forgeries: are such that the forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6^\star, r_i^\star)$ contains a triple $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star)$ for which $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star) \neq (\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3})$ for all $j \in [1, q]$.
- Type II F-forgeries: are such that $\sigma^\star$ contains $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star)$ such that $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star) = (\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3})$, for some $j \in [1, q]$ but $\sigma_6^\star \neq \sigma_{j,6}$.
- Type III F-forgeries: are such that $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6^\star, r^\star)$ contains $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_6^\star)$ such that $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_6^\star) = (\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \sigma_{j,6})$, for some $j \in [1, q]$, but the condition b.(ii) of definition 7 is satisfied. Namely, we are faced with one of the following situations.
  - Type III-A attacks: the predicate $\mathsf{Pred}^\star$ is of the form $\mathsf{Pred}^\star = (R^{\mathsf{EQ}}, i, \{F(x_t^\star)\}_{t \in \Upsilon})$, for some $i \in \Upsilon \subset \{1, \ldots, n\}$ and some vector $\vec{X}^\star = (x_1^\star, \ldots, x_n^\star)$ such that $x_i^\star$ is *not* the message at the i-th coordinate of $\vec{m}_j$ and the forged witness $W^\star$ nevertheless satisfies $\mathsf{Witness\text{-}Verify}(\mathsf{pk}, R^{\mathsf{EQ}}, i, \vec{X}^\star, W^\star, \sigma) = 1$.
  - Type III-B attacks: $\mathsf{Pred}^\star = (R^{\neg\mathsf{EQ}}, i, \{F(x_t^\star)\}_{t \in \Upsilon})$, for some index $i \in \Upsilon \subset \{1, \ldots, n\}$ and some vector $\vec{X}^\star = (x_1^\star, \ldots, x_n^\star)$ such that $x_i^\star$ *does* coincide with the message at the i-th coordinate of $\vec{m}_j$ although the witness $W^\star$ is such that $\mathsf{Witness\text{-}Verify}(\mathsf{pk}, R^{\neg\mathsf{EQ}}, i, \vec{X}^\star, W^\star, \sigma) = 1$.
  - Type III-C attacks: $\mathsf{Pred}^\star = (R^{\mathsf{IP}}, 0, \vec{X}^\star)$, for some vector $\vec{X}^\star \in \mathbb{Z}_p^n$ and the witness $W^\star$ satisfies $\mathsf{Witness\text{-}Verify}(\mathsf{pk}, R^{\mathsf{IP}}, i, \vec{X}^\star, W^\star, \sigma^\star) = 1$ whereas $\vec{m}_j \cdot \vec{X}^\star \neq 0$.
  - Type III-D attacks: $\mathsf{Pred}^\star = (R^{\neg\mathsf{IP}}, 0, \vec{X}^\star)$, for some vector $\vec{X}^\star \in \mathbb{Z}_p^n$, and $W^\star$ is such that $\mathsf{Witness\text{-}Verify}(\mathsf{pk}, R^{\neg\mathsf{IP}}, 0, \vec{X}^\star, W^\star, \sigma^\star) = 1$ while $\vec{m}_j \cdot \vec{X}^\star = 0$.

We observe that Type I and Type II F-forgeries encompass those for which the adversary wins the game of definition 7 because $\sigma^\star$ was not the output of any signing query. Type III attacks cover the case of the adversary winning the game because condition b.(ii) is satisfied in definition 7.

Type I forgeries are easily seen (see lemma 1) to break the HSDH assumption whereas lemma 2 and lemma 3 show that Type II and Type III forgeries give rise to algorithms solving the FlexDH and FlexDHE problems, respectively. □

Lemma 1. *Any Type I F-forgery contradicts the q-HSDH assumption, where q is the number of signing queries.*

Proof. The proof is based on ideas from [11]. We show a simple algorithm $\mathcal{B}$ that, on input of $(g, u, \Omega = g^\omega) \in \mathbb{G}^3$, and a set tuples $(A_i = g^{1/(\omega + c_i)}, B_i = g^{c_i}, C_i = u^{c_i})$ with $c_1, \ldots, c_q \in_R \mathbb{Z}_p$, uses a Type I forger $\mathcal{A}$ to find a triple $(g^{1/(\omega+c)}, g^c, u^c)$ such that $c \neq c_i$ for $i = 1, \ldots, q$. To generate the public key pk, $\mathcal{B}$ chooses $\beta_0, \beta_1 \xleftarrow{R} \mathbb{Z}_p$, $\alpha, \gamma \xleftarrow{R} \mathbb{Z}_p^*$ and sets $U_0 = g^{\beta_0}$, $U_1 = g^{\beta_1}$, $U_0 = g^{\beta_0}$,

$U_1 = g^{\beta_1}$, $A = g^{\gamma}$. It also defines $\{g_i\}_{i \in [1,2n] \setminus \{n+1\}}$ using a random $\alpha \xleftarrow{R} \mathbb{Z}_p$. The public key $\mathsf{pk} = \big(g, h, \Omega, A, U_0, U_1, \{g_i\}_{i \in [1,2n] \setminus \{n+1\}}\big)$ is given to $\mathcal{A}$.

To answer a signing query involving a vector $\vec{m} = (m_1, \ldots, m_n)$, $\mathcal{B}$ first picks $r \xleftarrow{R} \mathbb{Z}_p$, computes $\sigma_6 = V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$ and defines the polynomial $P_{\vec{m}}[X] = r + \sum_{j=1}^{n} m_{n+1-j} X^j$. It then generates signature parts $(\sigma_1, \sigma_2, \sigma_3)$ as (A_k^{γ}, B_k, C_k) using the next available tuple (A_k, B_k, C_k) (with $k \in [1, q]$). As for signature elements $\sigma_4 = (U_0 \cdot V^{\beta_1})^{c_k}$ and $\sigma_5 = V^{c_k}$, they are calculated as $\sigma_4 = {B_k}^{\beta_0 + \beta_1 \cdot P_{\vec{m}}(\alpha)}$ and $\sigma_5 = {B_k}^{P_{\vec{m}}(\alpha)}$. The game ends with $\mathcal{A}$ outputting a Type I forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6^\star, r^\star)$. Since $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star)$ did not appear in any signing query, $({\sigma_1^\star}^{1/\gamma}, \sigma_2^\star, \sigma_3^\star)$ must solve the q-HSDH instance. □

Lemma 2. *A Type II F-forger with success probability ε allows breaking the FlexDH assumption with probability ε/q, where q is the number of signing queries.*

Proof. The proof is inspired from [24][Lemma 2]. From a Type II forger, we build an algorithm $\mathcal{B}$ that finds a non-trivial triple $(g^{\mu}, g^{\mu a}, g^{\mu a b})$ on input of $(g, g_a = g^a, g_b = g^b)$. To prepare the public key pk, $\mathcal{B}$ chooses $\omega \xleftarrow{R} \mathbb{Z}_p^*$, $\gamma_u, \gamma_a \xleftarrow{R} \mathbb{Z}_p^*$ and sets $\Omega = g^{\omega}$, $A = (g_a \cdot g^{\omega})^{\gamma_a}$ (so that $\gamma = \log_g(A)$ is implicitly defined as $\gamma = (a + \omega)\gamma_a$) and $u = g^{\gamma_u}$. Next, $\mathcal{B}$ picks two pairs $(\tau_0, \tau_1) \xleftarrow{R} (\mathbb{Z}_p)^2$, $(\rho_0, \rho_1) \xleftarrow{R} (\mathbb{Z}_p)^2$ and defines $U_0 = g^{\rho_0} \cdot g_b^{\tau_0}$, and $U_1 = g^{\rho_1} \cdot g_b^{\tau_1}$. This implicitly defines the values $\beta_0 = \log_g(U_0) = \rho_0 + b\tau_0$ and $\beta_1 = \rho_1 + b\tau_1$, which are not available to $\mathcal{B}$. As in the proof of lemma 1, $\{g_i\}_{i \in [1,2n] \setminus \{i\}}$ are also defined for a randomly chosen $\alpha \xleftarrow{R} \mathbb{Z}_p$. At the outset of the game, $\mathcal{B}$ also chooses $q^\star \xleftarrow{R} [1, q]$.

When the signature of a vector $\vec{m} = (m_1, \ldots, m_n)$ is queried, the query's treatment depends on the index $k \in [1, q]$ of the query.

- If $k \neq q^\star$, $\mathcal{B}$ picks $r \xleftarrow{R} \mathbb{Z}_p$ and computes $\sigma_6 = V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$ and defines the polynomial $P_{\vec{m}}[X] = r + \sum_{j=1}^{n} m_{n+1-j} X_j$. It generates the triple $(\sigma_1, \sigma_2, \sigma_3)$ by setting $\sigma_1 = (g_a \cdot g^{\omega})^{\gamma_a/(\omega + c_k)}$, $\sigma_2 = g^{c_k}$, $\sigma_3 = u^{c_k}$ for a randomly drawn $c_k \xleftarrow{R} \mathbb{Z}_p$. Since it knows $\log_g(V) = P_{\vec{m}}(\alpha)$, it can also compute $\sigma_4 = (g^{\rho_0 + P_{\vec{m}}(\alpha) \cdot \rho_1} \cdot (g_b)^{\tau_0 + P_{\vec{m}}(\alpha) \cdot \tau_1})^{c_k}$ and $\sigma_5 = g^{P_{\vec{m}}(\alpha) \cdot c_k}$,
- If $k = q^\star$, $\mathcal{B}$ will implicitly define $c_{q^\star} = a$ by setting $\sigma_1 = g^{\gamma_a}$, $\sigma_2 = g_a$, $\sigma_3 = g_a^{\gamma_u}$. Next, $\mathcal{B}$ considers the polynomial $P_0[X] = \sum_{j=1}^{n} m_{n+1-j} X^j$ and defines $r = -P_0(\alpha) - \tau_0/\tau_1$ and computes $\sigma_6 = V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$. We note that, since τ_0, τ_1 are independent of $\mathcal{A}$'s view, the above choice of r gives a random-looking σ_6. Moreover, we have $U_0 \cdot V^{\beta_1} = g^{\rho_0 + P_{\vec{m}}(\alpha)\rho_1}$, where $P_{\vec{m}}[X] = r + P_0[X]$, so that σ_4 is computable as $\sigma_4 = (g_a)^{\rho_0 + P_{\vec{m}}(\alpha)\rho_1}$. Finally, $\sigma_5 = (g^a)^{P_{\vec{m}}(\alpha)}$ is also easily computable.

Finally, $\mathcal{A}$ outputs a forgery $\sigma^\star$ such that $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star)$ appeared in the output of some signing query. With probability $1/q$, this query happens to be the $q^\star$-th query (and $\mathcal{B}$ fails otherwise), so that $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star) = (g^{\gamma_a}, g_a, g_a^{\gamma_u})$. By assumption, the output of this query contained an element $\sigma_{q^\star,6}$ such that $\sigma_{q^\star,6} \neq \sigma_6^\star$. We call $\chi_0 = r + P_0(\alpha)$ the value $\chi_0 = \log_g(\sigma_{q^\star,6})$. The fourth component of the forgery can be written as

$$\sigma_4^\star = (g^a)^{\rho_0 + \chi\rho_1} \cdot (g^{ab})^{\tau_0 + \chi\tau_1}$$

where $\chi = \log_g(\sigma_6^\star)$. Since the $q^\star$-th query involved the same $c_{q^\star} = a$, dividing out the value $\sigma_{q^\star,4} = (g_a)^{\rho_0 + P_{\vec{m}}(\alpha)\rho_1} = (g^a)^{\rho_0 + \chi_0\rho_1} \cdot (g^{ab})^{\tau_0 + \chi_0\tau_1}$ from $\sigma_4^\star$ yields

$$T_4 = \sigma_4^\star / \sigma_{q^\star,4} = (g^a)^{(\chi - \chi_0)\rho_1} \cdot (g^{ab})^{(\chi - \chi_0)\tau_1}$$

whereas the quotient of $\sigma_5 = h^{\chi_0 a}$ and $\sigma_5^\star$ reveals $T_5 = \sigma_5^\star/\sigma_5 = h^{(\chi-\chi_0)a}$. Hence, $\mathcal{B}$ extracts

$$R_3 = \left(g^{(\chi-\chi_0)}\right)^{ab} = T_4/T_5, \qquad R_2 = \left(g^{(\chi-\chi_0)}\right)^{a} = T_5, \qquad R_1 = \frac{\sigma_6^\star}{\sigma_{q^\star,6}} = g^{(\chi-\chi_0)}$$

which must form a non-trivial triple $(g^\mu, g^{\mu a}, g^{\mu ab}) = (R_1, R_2, R_3)$, with $\mu = \chi - \chi_0$. □

Lemma 3. *Any PPT adversary outputting a Type III F-forgery would contradict the n-FlexDHE assumption.*

Proof. We outline an algorithm $\mathcal{B}$ that takes as input a tuple $(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$, where $g_\ell = g^{(\alpha^\ell)}$ for each $\ell \in [1, 2n]\backslash\{n+1\}$ and uses a Type III forger $\mathcal{A}$ to find a non-trivial tuple $(g^\mu, g_{n+1}^\mu, g_{2n}^\mu) = (g^\mu, g^{\mu\cdot(\alpha^{n+1})}, g^{\mu\cdot(\alpha^{2n})})$.

Algorithm $\mathcal{B}$ uses its input to define $\{g_\ell\}_{\ell\in[1,2n]}$ whereas other components of the public key are chosen as specified by the key generation algorithm and $\mathcal{B}$ retains the private key $\mathsf{sk} = (\gamma, \omega, \beta_1)$ which allows perfectly answering signing queries.

Type III-A attacks: We first consider Type III-A forgeries. By assumption, $\mathcal{A}$ outputs a forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6^\star, r^\star)$ such that $\sigma_6^\star$ appeared in the j-th signing query, for some $j \in [1, q]$. The adversary also outputs a predicate $\mathsf{Pred}^\star = (R^{\mathsf{EQ}}, i, \{F(x_t^\star)\}_{t\in\Upsilon})$ consisting of an index $i \in [1, n]$, a set of function values $\{F(x_t^\star) = (F_{t,1}, F_{t,2}, F_{t,3}) = (g_1^{x_t^\star}, g^{x_t^\star}, g_{2n}^{x_t^\star})\}_{t\in\Upsilon}$, for some vector $\vec{X}^\star = (x_1^\star, \ldots, x_n^\star)$ and some subset $\Upsilon \subset \{1, \ldots, n\}$, as well as a witness $W^\star \in \mathbb{G}$ such that $e(g_i, \sigma_6^\star) = e(g, W^\star) \cdot e(F_{i,1}, g_n)$. From the j-th signing query $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$ (for which $x_i^\star \neq m_{j,i}$ by assumption) made by the adversary, $\mathcal{B}$ is able to construct a witness $W' \in \mathbb{G}$ such that $e(g_i, \sigma_6^\star) = e(g, W') \cdot e(g_1, g_n)^{m_{j,i}}$ since it had to compute $\sigma_6^\star$ itself when answering the query. This implies that

$$e(g, W^\star/W') = e(g_1^{m_{j,i}-x_i^\star}, g_n) = e(g_1^{m_{j,i}} \cdot F_{i,1}^{-1}, g_n).$$

Since the right-hand-side member of the first equality equals $e(g_1^{m_{j,i}-x_i^\star}, g_n) = e(g, g_{n+1}^{m_{j,i}-x_i^\star})$, it easily comes that $W^\star/W' = g_{n+1}^{m_{j,i}-x_i^\star}$. Moreover, from the values $(F_{i,2}, F_{i,3})$, algorithm $\mathcal{B}$ also obtains $g^{m_{j,i}-x_i^\star} = g^{m_{j,i}} \cdot F_{i,2}^{-1}$ and $g_{2n}^{m_{j,i}-x_i^\star} = g_{2n}^{m_{j,i}} \cdot F_{i,3}^{-1}$ which yield a solution $(g^{m_{j,i}}/F_{i,2}, W^\star/W', g_{2n}^{m_{j,i}}/F_{i,3})$ to the n-FlexDHE instance with $\mu = m_{j,i} - x_i^\star$.

Type III-B attacks: We now turn to Type III-B forgeries. Namely, at the end of the game, the adversary $\mathcal{A}$ outputs a forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6^\star, r^\star)$ such that $\sigma_6^\star$ was part of the j^{th} signing query, for some $j \in [1, q]$. The adversary's output $(\mathsf{Pred}^\star, W^\star)$ also includes an index $i \in [1, n]$, a set of function values $\{F(x_t^\star) = (F_{t,1}, F_{t,2}, F_{t,3}) = (g_1^{x_t^\star}, g^{x_t^\star}, g_{2n}^{x_t^\star})\}_{t\in\Upsilon}$ and a witness $W^\star = (W_0^\star, W_1^\star, W_2^\star, W_3^\star, W_4^\star) \in \mathbb{G}^5$ such that $W_1^\star \neq 1_{\mathbb{G}}$ and

$$e(g_i, \sigma_6^\star) \cdot e(F_{i,1}, g_n)^{-1} = e(g_i, \sigma^\star \cdot g_{n+1-i}^{-x_i^\star}) = e(W_1^\star, g_n) \cdot e(g, W_4^\star) \tag{23}$$

$$e(W_1^\star, g) = e(g_1, W_2^\star) \tag{24}$$

$$e(W_1^\star, g_{2n}) = e(g_1, W_3^\star), \tag{25}$$

which would trick a verifier into believing that $m_{j,i} \neq x_i^\star$ although $m_{j,i} = x_i^\star$. From the j-th signing query, $\mathcal{B}$ knows the signed vector $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$ as well as an exponent $r_j \in \mathbb{Z}_p$ such that $\sigma_6^\star = g^{r_j} \cdot \prod_{k=1}^n g_{n+1-k}^{m_{j,k}}$. By hypothesis, we have $m_{j,i} = x_i^\star$, which allows $\mathcal{B}$ to compute $W' = g_i^{r_j} \cdot \prod_{k=1}^n g_{n+1-k+i^\star}^{m_{j,k}}$ such that

$$e(g_i, \sigma_6^\star \cdot g_{n+1-i}^{-x_i^\star}) = e(g, W'). \tag{26}$$

The combination of (23) and (26) implies $e(g, W'/W_4^\star) = e(W_1^\star, g_n)$. If we define $\mu = \log_{g_1}(W_1^\star)$, the latter relation implies that $W'/W_4^\star = g_{n+1}^{\mu}$ and equations (24)-(25) guarantee that we also have $W_2^\star = g^{\mu}$ and $W_3^\star = g_{2n}^{\mu}$. It comes that

$$\big(W_2^\star, W'/W_4^\star, W_3^\star\big) = (g^{\mu}, g_{n+1}^{\mu}, g_{2n}^{\mu})$$

forms a non-trivial solution to the n-FlexDHE instance.

Type III-C attacks: We now assume that the adversary $\mathcal{A}$ produces a Type III-C forgery and show that it allows computing $g_{n+1} = g^{(\alpha^{n+1})}$, and *a fortiori* break n-FlexDHE.

The game ends with the adversary $\mathcal{A}$ outputting a forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6^\star, r^\star)$ that coincides with the output of the j-th signing query, for some index $j \in [1, q]$. The adversary $\mathcal{A}$ also outputs a witness $W^\star \in \mathbb{G}$ and a predicate $\mathsf{Pred}^\star = (R^{\mathsf{IP}}, 0, \vec{X}^\star)$ consisting of a vector $\vec{X}^\star = (x_1^\star, \ldots, x_n^\star)$ such that $\vec{m}_j \cdot \vec{X}^\star \neq 0$. Yet, since $\mathsf{Witness\text{-}Verify}(\mathsf{pk}, R^{\mathsf{IP}}, 0, \vec{X}^\star, W^\star, \sigma^\star) = 1$, it must hold that

$$e\big(\prod_{i=1}^{n} g_i^{x_i^\star}, \sigma_6^\star\big) = e(g, W^\star). \tag{27}$$

From the j-th signing query $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$, $\mathcal{B}$ knows an opening $(m_{j,1}, \ldots, m_{j,n}; r_j)$ of the commitment $\sigma_6^\star$ (*i.e.*, $\sigma_6^\star = \mathrm{VecCom}(\vec{m}_j; r_j)$). This allows computing $\tilde{W}_k = g_k^{r_j} \cdot \prod_{\ell=1,\ell\neq k}^{n} g_{n+1-\ell+k}^{m_{j,\ell}}$ for $k = 1$ to n, which in turn yields $\tilde{W} = \prod_{k=1}^{n} \tilde{W}_k^{x_k^\star} \in \mathbb{G}$ such that

$$e\big(\prod_{i=1}^{n} g_i^{x_i^\star}, \sigma_6^\star\big) = e(g_1, g_n)^{\vec{m}_j \cdot \vec{X}^\star} \cdot e(g, \tilde{W}). \tag{28}$$

Combining (27) and (28), we find $e(g, W^\star/\tilde{W}) = e(g_1, g_n)^{\vec{m}_j \cdot \vec{X}^\star}$, so that $g_{n+1} = (W^\star/\tilde{W})^{\frac{1}{\vec{m}_j \cdot \vec{X}^\star}}$ and *a fortiori* break the n-FlexDHE assumption.

Type III-D attacks: We are left with Type III-D forgeries where the game ends with the adversary $\mathcal{A}$ outputting a forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star, \sigma_5^\star, \sigma_6^\star, r^\star)$ that coincides with the output of the j-th signing query, for some $j \in [1, q]$, but $\mathcal{A}$ also produces a vector $\mathsf{Pred}^\star = (R^{\neg\mathsf{IP}}, 0, \vec{X}^\star)$ for some vector $\vec{X}^\star = (x_1^\star, \ldots, x_n^\star)$ such that $\vec{m}_j \cdot \vec{X}^\star = 0$ and a witness $W^\star$ that would wrongly convince a verifier that $\vec{m}_j \cdot \vec{X}^\star \neq 0$. The adversary's output thus comprises $W^\star = (W_0^\star, W_1^\star, W_2^\star, W_3^\star, W_4^\star) \in \mathbb{G}^5$ such that $W_1^\star \neq 1_{\mathbb{G}}$ and

$$e\big(\prod_{i=1}^{n} g_i^{x_i^\star}, \sigma_6^\star\big) = e(W_1^\star, g_n) \cdot e(g, W_4^\star) \tag{29}$$

$$e(W_1^\star, g) = e(g_1, W_2^\star) \tag{30}$$

$$e(W_1^\star, g_{2n}) = e(g_1, W_3^\star) \tag{31}$$

From the j-th signing query, $\mathcal{B}$ has recollection of the vector $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$ and some $r_j \in \mathbb{Z}_p$ such that $(m_{j,1}, \ldots, m_{j,n}; r_j)$ forms an opening of $\sigma_6^\star = \mathrm{VecCom}(\vec{m}_j; r_j) = g^{r_j} \cdot \prod_{k=1}^{n} g_{n+1-k}^{m_{j,k}}$. By hypothesis, it holds that $\vec{m}_j \cdot \vec{X}^\star = 0$, so that $\mathcal{B}$ can use $(\vec{m}_j; r_j)$ and $\vec{X}^\star$ to compute $\tilde{W} \in \mathbb{G}$ such that

$$e(\prod_{i=1}^{n} g_i^{x_i^\star}, \sigma_6^\star) = e(g, \tilde{W}). \tag{32}$$

Then, (29) and (32) guarantee that $e(g, \tilde{W}/W_4^\star) = e(W_1^\star, g_n)$. Now, if we define $\mu = \log_{g_1}(W_1^\star)$ (which is non-zero since $W_1^\star \neq 1_{\mathbb{G}}$), we thus have $\tilde{W}/W_4^\star = g_{n+1}^\mu$ and equations (30)-(31) in turn imply $W_2^\star = g^\mu$ and $W_3^\star = g_{2n}^\mu$. It eventually comes that

$$\big(W_2^\star, \tilde{W}/W_4^\star, W_3^\star\big) = (g^\mu, g_{n+1}^\mu, g_{2n}^\mu)$$

is a non-trivial solution to the given n-FlexDHE instance. □

Remark 1. In the proof of lemma 3, the reduction needs to know the vector $\vec{X}^\star = (x_1^\star, x_2^\star, \ldots, x_n^\star)$ in the case of Type III-C and Type III-D forgeries because its coordinates are needed to construct the witnesses $\tilde{W}$ that satisfy (28) in the Type III-C case and (32) in Type III-D forgeries. In these cases, the proof of lemma 3 ceases to go through if the reduction only knows[3] $F(x_j^\star)$ for $j = 1$ to n.

Nevertheless, it is easy to see that the proof of lemma 3 still works when $\mathcal{B}$ is only given $\{F(x_j^\star)\}_{j=1}^n$ for values $\{x_j^\star\}_{j=1}^n$ that are somehow (*i.e.*, by means of additional proofs) constrained to belong to a small polynomial-sized interval. In this case, $\mathcal{B}$ can determine $\{x_j^\star\}_{j=1}^n$ from $\{F(x_j^\star)\}_{j=1}^n$ by applying $F(.)$ to all interval elements. This fact is useful when certain threshold predicates about the vector $\vec{m}$ are proven as noted in section E.4.

C Proof of Theorem 2

Signer Privacy: In an execution of SigIssue $\leftrightarrows$ SigObtain, the simulator SimIssue receives a commitment $C_{\vec{m}_{|n_1}}$ of some vector $\vec{m}_{|n_1}$ and a public vector $(m_{n_1+1}, \ldots, m_n)$. The simulator first rewinds the adversary from the WI proof of knowledge so as to extract $\vec{m}_{|n_1}, open_{\vec{m}_{|n_1}}$, where $\vec{m}_{|n_1} = (m_1, \ldots, m_{|n_1})$ and $open_{\vec{m}_{|n_1}} = (\vec{m}_{|n_1}, r')$. If the extraction fails, it outputs $\perp$. Otherwise, it queries $\vec{m} = (m_1, \ldots, m_{n_1}, m_{n_1+1}, \ldots, m_n)$ to its signing oracle and obtains a signature $\sigma = (\sigma_1, \ldots, \sigma_6, r)$ and sends σ as well as $r'' = r - r'$ back to the adversary. It is easy to see that, unless SimIssue fails to extract $open_{\vec{m}_{|n_1}}$ using the knowledge extractor, the simulation is perfect.

User Privacy: The simulator now has to emulate the user running SigObtain in an execution of SigIssue $\leftrightarrows$ SigObtain without using the user's private inputs $(\vec{m}_{|n_1}, open_{\vec{m}_{|n_1}})$. To this end, the user-simulator SimObtain uses the simulator of the interactive WI proof of knowledge of an opening of the commitment $C_{\vec{m}_{|n_1}}$. The simulator is trivial since, given that the commitment is perfectly hiding, any opening gives identically distributed proofs.

D Proof of Theorem 3

We show that, when the reference string $\mathbf{f} = (\vec{f_1}, \vec{f_2}, \vec{f_3})$ produced by SimSetup is prepared for the WI setting (*i.e.*, with $\vec{f_3} = \vec{f_1}^{\xi_1} \odot \vec{f_2}^{\xi_2} \odot (1, 1, g)^{-1}$), all proofs produced by these algorithms can be simulated without knowing any witness using the trapdoor of the simulated CRS.

More precisely, the simulator Sim = (SimSetup, SimSigProve$_1$, SimSigProve$_2$, SimEqComProve$_1$, SimEqComProve$_2$) proceeds as follows.

[3] This is actually the main reason why the model considers two families of relations $\mathcal{R}_1$ and $\mathcal{R}_2$ rather than one: $\mathcal{R}_1$ is the family of relations for which the coordinates of $\vec{X}$ are only available as $\{F(x_i)\}_{i \in \Upsilon}$ whereas $\mathcal{R}_2$ is a family of relations for which we consider $\vec{X} = (x_1, \ldots, x_n)$ to be public.

SimSetup: the Groth-Sahai CRS $\mathbf{f} = (\vec{f_1}, \vec{f_2}, \vec{f_3})$ is chosen so as to have $\vec{f_3} = \vec{f_1}^{\xi_1} \odot \vec{f_2}^{\xi_2} \odot (1,1,g)^{-1}$ for some $\xi_1, \xi_2 \in \mathbb{Z}_p$. The pair $\tau = (\xi_1, \xi_2)$ will serve as a trapdoor allowing to simulate proofs without knowing any witnesses.

SimSigProve$_1$: on a simulated CRS, the witnesses $\{\sigma_i = 1_{\mathbb{G}}\}_{i=1}^{7}$ and can be used to generate a proof for equations (8)-(9) of π_σ.

- In the case $R = R^{\mathsf{EQ}}$, the assignment $X_{i,1} = X_{i,2} = X_{i,3} = W = 1_{\mathbb{G}}$ yields a proof π_{x_i}, $\{\pi_{X_t,j}\}_{t \in \Upsilon, j \in \{1,2\}}$ for equation (11)-(12).
- In the case, $R = R^{\neg\mathsf{EQ}}$, together with the trivial assignment $\sigma_6 = 1_{\mathbb{G}}$, the witnesses $X_{i,1} = g_1^{-1}$, $W_1 = g_1$, $W_2 = g$, $W_3 = g_{2n}$ and $W_4 = 1_{\mathbb{G}}$ provide valid proofs π_{x_i}, π_W and $\{\pi_{X_t,j}\}_{t \in \Upsilon, j \in \{1,2\}}$ for equations (13)-(14) and (12), respectively.

As for relations (10) of the proof π_σ, the NIZK simulator can make use of the trapdoor τ, which allows trapdoor opening to 0 a commitment C_{θ_1} to $\theta_1 = 1$ (so as to generate a fake proof that $\sigma_7 = A$ in the last relation of (10)).

SimSigProve$_2$: the witnesses $\{\sigma_i = 1_{\mathbb{G}}\}_{i=1}^{7}$ and $\Theta = W = 1_{\mathbb{G}}$ can be used to generate proofs for equations (8)-(9) of π_σ.

- If $R = R^{\mathsf{IP}}$, the witness $W = 1_{\mathbb{G}}$ gives a proof for equation of (15).
- If $R = R^{\neg\mathsf{IP}}$, the assignment $\Theta = W_0 = W_1 = W_2 = W_3 = W_4 = 1_{\mathbb{G}}$ gives us proofs for relations (16), (18) and the leftmost equation of (17). As for the last equations of (10) and (17), the NIZK simulator uses the trapdoor τ of the CRS to generate proofs for the false statements $\sigma_7 = A$ and $\Theta = g$ (by trapdoor opening to 0 the commitment to $\theta_1 = 1$).
- If $R = R^{\mathsf{EQ}}$, setting $X_i = W = 1_{\mathbb{G}}$ (along with $\sigma_6 = 1_{\mathbb{G}}$) allows simulating a proof for the first equation of (19) whereas the second relation of (19) is handled (like the last equation of (10)) by trapdoor opening to 0 the commitment to $\theta_1 = 1$ (thanks to the trapdoor τ).
- If $R = R^{\neg\mathsf{EQ}}$, we can set $\Theta = X_i = W_0 = W_1 = W_2 = W_3 = W_4 = 1_{\mathbb{G}}$ so as to satisfy (20)-(21) and use the trapdoor τ to fake a proof that $X_i = g_1^{x_i}$ and $\Theta = g$ (in the two equations of (22)) at the same time as the proof that $\sigma_7 = A$ (in the last equation of (10)).

SimEqComProve: recall that $\mathsf{EqComProve}$ aims at proving that $\vec{C_X}$ and $\vec{C_Y}$ are Groth-Sahai commitments to the same $X = Y \in \mathbb{G}$. To this end, it uses the random coins that were used to compute $\vec{C_X} = (1,1,X) \odot \vec{f_1}^{r_x} \odot \vec{f_2}^{s_x} \odot \vec{f_3}^{t_x}$ and $\vec{C_Y} = (1,1,Y) \odot \vec{f_1}^{r_y} \odot \vec{f_2}^{s_y} \odot \vec{f_3}^{t_y}$ to define variables $(\rho_1, \rho_2, \rho_3) = (r_x - r_y, s_x - s_y, t_x - t_y) \in (\mathbb{Z}_p)^3$ (and their commitments $\vec{C_{\rho_1}}, \vec{C_{\rho_2}}, \vec{C_{\rho_3}}$) such that

$$\vec{C_X} \odot \vec{C_Y}^{-1} = \left(f_1^{\rho_1} \cdot f_{31}^{\rho_3},\ f_2^{\rho_2} \cdot f_{32}^{\rho_3},\ g^{\rho_1 + \rho_2} \cdot f_{33}^{\rho_3}\right), \tag{33}$$

where $\vec{f_1} = (f_1, 1, g)$, $\vec{f_2} = (1, f_2, g)$ and $\vec{f_3} = (f_{31}, f_{32}, f_{33})$. On a simulated CRS, commitments $\{\vec{C_{\rho_j}}\}_{j \in \{1,2,3\}}$ can be replaced by commitments to 0 and the trapdoor τ makes it possible to generate fake a proof for equation (33) (see, e.g., [14][Section 4.4] for details).

On a simulated (*i.e.*, witness indistinguishable) CRS, all commitments are perfectly hiding and simulated proofs are perfectly indistinguishable from real proofs since proofs always have the same distribution, no matter which witnesses are used to generate them.

Moreover, since perfectly WI and perfectly sound common reference strings are indistinguishable under the DLIN assumption, no PPT adversary can tell apart real proofs and simulated proofs as long as the DLIN assumption holds in $\mathbb{G}$. □

E Applications of Block-Wise P-signatures to Anonymous Credentials

Anonymous credential systems involve users, organizations (which deliver credentials to users) and a certification authority (CA) which is responsible for registering users' public keys.

E.1 Non-Interactive Anonymous Credentials with Efficient Attributes

Let $n \in \mathsf{poly}(\lambda)$. Let $\mathcal{R}_1, \mathcal{R}_2 : [0, n] \times \mathcal{D}^n \times \mathcal{D}^n \to \{0,1\}$ be relation families, for some domain $\mathcal{D}$, and F be an injective function. A *non-interactive* anonymous credential system *with efficient attributes* consists of:

- a vector commitment scheme (VecCom, VecOpen).
- a set ($\mathsf{CredSetup}$, $\mathsf{PseudonymIssue}$, $\mathsf{PseudonymObtain}$, $\mathsf{CredIssue}$, $\mathsf{CredObtain}$, $\mathsf{CredProve}$, $\mathsf{CredVerify}$) of PPT algorithms or protocols with the following specifications.

$\mathsf{CredSetup}(\lambda)$: takes as input a security parameter λ and outputs at set of public parameters params. As part of this protocol, users and organizations generate their public and secret keys using auxiliary algorithms $\mathsf{OKeygen}$ and $\mathsf{UKeygen}$ which are described hereafter. We assume that they register their public keys with the CA. We refer to PKI as the collection of all public keys, and to the identity of the user as his public key pk. The outcome of the registration step is the user (whose private input is his secret key sk) obtaining a root credential C_{CA} whereby the CA authenticates sk.

$\mathsf{OKeygen}(\lambda)$: allows an organization O to generate its key pair $(\mathsf{sk}_O, \mathsf{pk}_O)$. We assume that, as part of its public key pk_O, organization O specifies an integer $n \in \mathsf{poly}(\lambda)$, which is the size of attribute vectors that O is willing to certify.

$\mathsf{UKeygen}(\lambda)$: is a user key generation algorithm outputting a pair $(\mathsf{sk}_U, \mathsf{pk}_U)$.

$\mathsf{PseudonymIssue}(\cdot) \leftrightarrows \mathsf{PseudonymObtain}(\mathsf{pk}_U, \mathsf{sk}_U, C_{CA})$: is an interactive protocol allowing a user and an organization to agree on a pseudonym (nym) N for the user. The users's private input is his key pair $(\mathsf{pk}_U, \mathsf{sk}_U)$ and his root credential C_{CA} whereas the organization does not have any private input. Their common output is the pseudonym N. The user has a private output aux_N consisting of some auxiliary information that may be needed later on.

$\mathsf{CredIssue}\big(\mathsf{pk}_O, \mathsf{sk}_O, N, V, (m_{n_1+1}, \ldots, m_n)\big) \rightleftarrows \mathsf{CredObtain}\big(\mathsf{pk}_U, \mathsf{sk}_U, N, \mathsf{aux}_N, \vec{m}_{|n_1}, (m_{n_1+1}, \ldots, m_n)\big)$: is an interactive protocol whereby a user obtains a credential from an organization without leaking his identity and while just revealing his pseudonym N. The user's input is $(\mathsf{pk}_U, \mathsf{sk}_U, \mathsf{aux}_N)$, a committed vector $\vec{m}_{|n_1} = (m_1, \ldots, m_{n_1})$, for some integer $n_1 \in [1, n]$, such that $m_1 = \mathsf{sk}_U$ and a public vector $(m_{n_1+1}, \ldots, m_n)$; the organization's private input is its key pair $(\mathsf{pk}_O, \mathsf{sk}_O)$, a commitment $V = \mathrm{VecCom}((m_1, \ldots, m_{n_1}); r)$ and a public vector $(m_{n_1+1}, \ldots, m_n)$. The user's private output is his credential C which he obtains without letting the organization learn anything about $\vec{m}_{|n_1} = (m_1, \ldots, m_{n_1})$.

$\mathsf{CredProve}(\mathsf{pk}_U, \mathsf{sk}_U, \mathsf{aux}_{N_1}, \mathsf{aux}_{N_2}, C, R, i, \vec{m}, \vec{X})$: is an algorithm allowing a user - known to an organization O_1 under pseudonym N_1 and to another organization O_2 under pseudonym N_2 - to prove to O_2 his possession of a credential C from O_1 on a vector $\vec{m} = (m_1, \ldots, m_n)$ such that $R(i, \vec{m}, \vec{X}) = 1$. The algorithm takes as input $(\mathsf{pk}_U, \mathsf{sk}_U, \mathsf{aux}_{N_1}, \mathsf{aux}_{N_2})$, the vector $\vec{m}$, the credential C, some relation $R \in \mathcal{R}_1 \cup \mathcal{R}_2$ and some vector $\vec{X} = (x_1, \ldots, x_n)$. The output consists of a proof π_C.

$\mathsf{CredVerify}(N_2, i, R, \pi_C, \vec{X})$: is an algorithm run by organization O_2. It outputs 1 if π_C convinces O_2 about the validity of the statement proven by $\mathsf{CredProve}$.

Such an anonymous credential system should provide anonymity and unforgeability. In short, the latter captures that: (1) each pseudonym corresponds to a well-defined identity; (2) if a user with pseudonym N manages to prove that he holds a credential from organization O for which the committed $\vec{m}$ satisfies $R(i, \vec{m}, \vec{X}) = 1$ for some $\vec{X} \in \mathcal{D}^n$, then it must be the case that organization O has indeed issued a credential to some pseudonym N' corresponding to the same identity as N. In addition, that credential must correspond to a vector $\vec{m}$ satisfying the aforementioned properties.

The notion of anonymity requires that, even if an adversary corrupts the CA and a subset of organizations, it cannot tell whether (a) it is interacting with real users who obtain credentials and prove properties about them as dictated by the adversary; (b) it is interacting with a simulator producing a simulated set of public parameters and emulates real users without using any witnesses depending on provers' identities.

The definition of unforgeability proceeds analogously to definition 9 in the context of P-signatures: we assume that the interactive protocol $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$ requires the user to provide an interactive proof of knowledge of his committed sub-vector $(m_1, \ldots, m_{n_1})$ and we call $\mathcal{E}^{\mathcal{A}}_{\mathsf{CredObtain}}$ the knowledge extractor of this proof of knowledge.

Definition 10. *A anonymous credential system with efficient attributes for relation families* $\mathcal{R}_1, \mathcal{R}_2$ *is* **unforgeable** *if there are efficient algorithms* $(\mathsf{ExtractSetup}, \mathsf{Extract})$ *such that (i)* Setup *and* ExtractSetup *have statistically close output distributions; (ii) no PPT algorithm* $\mathcal{A}$ *obtain non-negligible advantage in the following game.*

1. *The challenger runs* $\mathsf{params} \leftarrow \mathsf{ExtractSetup}(\lambda)$ *and* $(\mathsf{sk}_O, \mathsf{pk}_O) \leftarrow \mathsf{OKeygen}(\lambda)$ *and gives the public key* pk_O *(which defines an integer* $n \in \mathsf{poly}(\lambda)$*) to the adversary* $\mathcal{A}$*.*
2. *On at most* $q \in \mathsf{poly}(\lambda)$ *occasions,* $\mathcal{A}$ *starts an execution of the* $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$ *protocol and plays the role of a user interacting with the* CredIssue*-executing challenger. At each execution* $j \in [1, q]$*, the challenger appeals to the knowledge extractor* $\mathcal{E}^{\mathcal{A}}_{\mathsf{CredObtain}}$ *to extract* $\mathcal{A}$*'s committed vector* $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$ *(or, more precisely, its restriction* $(m_{j,1}, \ldots, m_{j,n_1})$ *where* $n_1 \in [1, n]$*). We call* C_j *the credential (i.e., the signature) obtained by* $\mathcal{A}$ *at the end of the* j*-th query.*
3. $\mathcal{A}$ *outputs a pseudonym* N_2*, a vector* $\vec{X} \in \mathcal{D}^n$*, an integer* $i \in [0, n]$*, a relation* $R \in \mathcal{R}_1 \cup \mathcal{R}_2$ *and a proof* π_C*. At this stage, the challenger computes* $F(\mathsf{sk}_U) = \mathsf{Extract}(N_2)$*, for some injective function* F*. The adversary* $\mathcal{A}$ *wins if* $\mathsf{CredVerify}(N_2, i, R, \pi_C, \vec{X}) = 1$ *and neither of the signed vectors* $\vec{m}_j = (m_{j,1}, \ldots, m_{j,n})$ *was such that* $m_{j,1} = \mathsf{sk}_U$ *and* $R(i, \vec{m}_j, \vec{X}) = 1$*.*

As usual, $\mathcal{A}$*'s advantage is defined to be its success probability, taken over all random coins.*

Definition 11. *A non-interactive anonymous credential system with efficient attributes provides* **anonymity** *if no PPT adversary can tell apart the two games described hereafter.*

Real: *The challenger generates public parameters* params *using* CredSetup*. The adversary* $\mathcal{A}$ *receives* params *and is allowed to invoke the following oracles on polynomially many occasions.*

$\mathcal{Q}_{\mathsf{UKeygen}}(U)$**:** *generates a key pair* $(\mathsf{sk}_U, \mathsf{pk}_U)$ *on behalf of user* U *and outputs* pk_U*. At most one such query is allowed for each user* U*.*

$\mathcal{Q}_{\mathsf{PseudonymIssue}}(U, O_j)$**:** *allows* $\mathcal{A}$*, acting as a dishonest organization* O_j*, to request the generation of a pseudonym for user* U*. If no public key was defined for* U *or no root credential* C_{CA} *was defined for* pk_U*, the oracle outputs* $\perp$*. Otherwise, the oracle runs algorithm* PseudonymObtain *using the input* $(\mathsf{pk}_U, \mathsf{sk}_U, C_{CA})$ *in an execution of* $\mathsf{PseudonymIssue} \leftrightarrows \mathsf{PseudonymObtain}$*. The* PseudonymIssue*-executing* $\mathcal{A}$ *is allowed to choose the public key* pk_{O_j}

of the dishonest organization but, if the same organization O_j is involved in subsequent queries, the same public key pk_{O_j} must be re-used by $\mathcal{A}$. If the protocol terminates, $\mathcal{A}$ obtains a pseudonym N for user U and $\mathcal{Q}_{\mathsf{PseudonymIssue}}(U)$ holds aux_N back from the adversary.

$\mathcal{Q}_{\mathsf{CredObtain}}(U, N, O_j, \vec{m}_{|n_1}, (m_{n_1+1}, \ldots, m_n))$ *: this oracle first checks if a key pair $(\mathsf{sk}_U, \mathsf{pk}_U)$ was created and if a pseudonym N was registered by organization O_j for user U. If not, the oracle returns $\perp$. Otherwise, it starts an execution of*

$$\mathcal{A}(\mathsf{params}, U, \mathsf{state}) \leftrightarrows \mathsf{CredObtain}(\mathsf{pk}, \mathsf{sk}, N, aux_N, \vec{m}_{|n_1}, (m_{n_1+1}, \ldots, m_n))$$

using the adversarially-chosen vectors $\vec{m}_{|n_1}$ and $(m_{n_1+1}, \ldots, m_n)$ (where n is fixed by O_j's public key and $n_1 \in [1, n]$ is chosen by $\mathcal{A}$). If the protocols successfully terminates, it stores the obtained credential C_f.

$\mathcal{Q}_{\mathsf{CredProve}}(\mathsf{params}, U, N_1, N_2, C_f, R, i, \vec{m}, \vec{X})$**:** *first checks if pseudonyms N_1 and N_2 have been registered for user pk_U by some organizations O_1 and O_2, respectively. If not, the oracle outputs $\perp$. Otherwise, it checks if O_1 has indeed delivered a credential C_f to user U for the vector $\vec{m} \in \mathcal{D}^n$. If not or if $R(i, \vec{m}, \vec{X}) = 0$, it aborts. Otherwise, the oracle responds by returning the output of* $\mathsf{CredProve}(\mathsf{pk}_U, \mathsf{sk}_U, aux_{N_1}, aux_{N_2}, C_f, R, i, \vec{m}, \vec{X})$.

After a number of queries, the adversary halts and outputs a bit $b \in \{0, 1\}$.

Ideal: *The challenger generates simulated parameters* params *and a simulation trapdoor τ by running an algorithm* $\mathsf{SimCredSetup}(\lambda)$. *The adversary receivers* params_s *and is allowed to query the same oracles as in the* **Real** *game. However, instead of answering $\mathcal{Q}_{\mathsf{PseudonymIssue}}$, $\mathcal{Q}_{\mathsf{CredProve}}$ and $\mathcal{Q}_{\mathsf{CredObtain}}$ queries using users' private keys and credentials, the challenger makes use of a simulator* $\mathsf{SimCred}(\tau)$ *that uses τ to answer all queries without knowing any user's private inputs. After a number of queries, the adversary halts and outputs a bit $b \in \{0, 1\}$.*

The adversary $\mathcal{A}$'s advantage is defined in the standard way, as the difference between the probabilities to have $b = 1$ in games **Real** *and* **Ideal**.

In the above definition, the adversary's interaction with oracles $\mathcal{Q}_{\mathsf{PseudonymIssue}}$ and $\mathcal{Q}_{\mathsf{CredObtain}}$ is restricted to be sequential since, in the **Ideal** game, the simulator must be able to rewind the adversary when these oracles are invoked.

E.2 Construction from Block-Wise P-Signatures

Let Σ = ($\mathsf{Setup}, \mathsf{SigSetup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Witness\text{-}Gen}, \mathsf{Witness\text{-}Verify}, \{\mathsf{SigProve}_i, \mathsf{EqComProve}_i\}_{i=1,2}$, $\mathsf{SigIssue}, \mathsf{SigObtain}$) be a secure block-wise P-signature for the relation families $\mathcal{R}_1 = \{R^{\mathsf{EQ}}, R^{\neg\mathsf{EQ}}\}$, $\mathcal{R}_2 = \{R^{\mathsf{IP}}, R^{\neg\mathsf{IP}}\}$ and let (VecCom, VecOpen) and ($\mathsf{Com}, \mathsf{Open}$) be the associated commitment schemes. The construction of anonymous credentials with efficient attributes goes as follows.

CredSetup(λ)**:** runs $\mathsf{params} \leftarrow \Sigma.\mathsf{Setup}(\lambda)$ and generates a commitment key for (VecCom, VecOpen).

OKeygen(λ)**:** chooses $n \in \mathsf{poly}(\lambda)$, runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{SigSetup}(\lambda, n)$ and sets $(\mathsf{sk}_O, \mathsf{pk}_O) = (\mathsf{sk}, \mathsf{pk})$.

UKeygen(λ)**:** uniformly picks a secret key sk_U in the domain $\mathcal{D}$ and computes $\mathsf{pk}_U = \mathsf{PublicKey}(\mathsf{sk}_U)$ using some appropriately defined function $\mathsf{PublicKey}(.)$.

PseudonymIssue$(\cdot) \leftrightarrows$ **PseudonymObtain**$(\mathsf{pk}_U, \mathsf{sk}_U, C_{CA})$**:** the user U generates his pseudonym N as a perfectly binding commitment $N = \mathsf{Com}(\mathsf{sk}_U, open_{\mathsf{sk}_U})$ to his secret key sk_U, for some appropriate opening $open_{\mathsf{sk}_U}$ and sets the auxiliary information aux_N to be $open_{\mathsf{sk}_U}$. In addition, U generates a proof that he holds a credential from the CA for this pseudonym (using the algorithm below). He finally sends N to the organization and proves knowledge of $(\mathsf{sk}_U, open_{\mathsf{sk}_U})$ using some non-malleable [36] designated verifier [35] interactive proof.

$\mathsf{CredIssue}$ $(\mathsf{pk}_O, \mathsf{sk}_O, N, V', (m_{n_1+1}, \ldots, m_n)) \leftrightarrows$ **$\mathsf{CredObtain}$**$(\mathsf{pk}_U, \mathsf{sk}_U, N, \mathsf{aux}_N, \vec{m}_{|_{n_1}}, (m_{n_1+1}, \ldots, m_n))$**:** the takes as input a commitment $V' = \mathrm{VecCom}(\vec{m}_{|_{n_1}}, \mathit{open}_{\vec{m}_{|_{n_1}}})$, for some $\mathit{open}_{\vec{m}_{|_{n_1}}}$. Then, U provides O with an interactive zero-knowledge proof that he knows how to open V' to a vector containing sk_U in its first coordinate and that N opens to the same sk_U. Finally, U and O run $\mathsf{SigIssue}(\mathsf{sk}_O, V', (m_{n_1+1}, \ldots, m_n)) \leftrightarrows \mathsf{SigObtain}(\mathsf{pk}_O, \vec{m}_{|_{n_1}}, \mathit{open}_{\vec{m}_{|_{n_1}}})$. As a result of this protocol, U obtains a credential C consisting of O's signature σ on the vector $\vec{m} = (m_1, \ldots, m_{n_1}, m_{n_1+1}, \ldots, m_n)$ without revealing anything about $\vec{m}_{|_{n_1}}$.

$\mathsf{CredProve}$ $(\mathsf{pk}_U, \mathsf{sk}_U, \mathsf{aux}_{N_1}, \mathsf{aux}_{N_2}, C, R, i, \vec{m}, \vec{X})$**:** parse the credential C as a P-signature σ. Compute $(C_{x_1}, C_{W_1}, C_\sigma, \pi_1) \leftarrow \mathsf{SigProve}_1(\mathsf{pk}, R^{\mathsf{EQ}}, 1, \Upsilon = \{1\}, \sigma, \vec{m}, \vec{X}')$, where $\vec{X}' = (\mathsf{sk}_U, 0, \ldots, 0)$ and denote by C_{x_1} the commitment to $x_1 = \mathsf{sk}_U$ which is part of $\mathsf{SigProve}_1$'s output. Compute $(C_W, C'_\sigma, \pi_2) \leftarrow \mathsf{SigProve}_2(\mathsf{pk}, R, i, \sigma, \vec{m}, \vec{X})$. Finally, generate $\pi_3 \leftarrow \mathsf{EqComProve}(C_{x_1}, N_2, \mathsf{aux}_{N_2})$ as a proof that perfectly binding commitments N_2 and C_{x_1} open to the same value and a similar proof $\pi_4 \leftarrow \mathsf{EqComProve}(C_\sigma, C_{\sigma'})$ that C_σ and C'_σ conceal the same value. Return the complete proof $\pi_C = (C_{x_1}, C_{W_1}, C_\sigma, \pi_1, C_W, C'_\sigma, \pi_2, \pi_3, \pi_4)$.

$\mathsf{CredVerify}$ $(N_2, i, R, \pi_C, \vec{X})$**:** parse π_C as $(C_{x_1}, C_{W_1}, C_\sigma, \pi_1, C_{W_1}, C'_\sigma, \pi_2, \pi_3, \pi_4)$. Return 1 if it holds that $\mathsf{VerifyProof}_1(\mathsf{pk}, R^{\mathsf{EQ}}, 1, \pi_1, C_\sigma, C_W, \{C_{x_1}\}) = \mathsf{VerifyProof}_2(\mathsf{pk}, R, i, \pi_2, C'_\sigma, C_W, \vec{X}) = 1$ and if π_3 (resp. π_4) is a valid proof that C_{x_1} and N_2 (resp. C_σ and $C_{\sigma'}$) open to the same message. Otherwise, return 0.

We note that, if C_σ consists of several commitments to group elements, π_4 must be obtained by running $\mathsf{EqComProve}$ for each commitment that C_σ, C'_σ consist of.

When the above construction is instantiated using the concrete P-signature of section 3, the anonymous credential system can be optimized by only including C_σ (instead of both C_σ and C'_σ) in π_C and removing π_4.

Theorem 4. *The above anonymous credential system is unforgeable if the interactive proof used in* $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$ *is sound and if the underlying P-signature is itself unforgeable.*

Proof. We construct a reduction $\mathcal{B}$ that breaks the security of the P-signature in the sense of definition 8 using its interaction with an adversary $\mathcal{A}$ in the game of definition 10.

Whenever $\mathcal{A}$ decides to run an execution of $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$ (and thereby sends a pseudonym N and a commitment V to $\mathcal{B}$), $\mathcal{B}$ uses the knowledge extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{CredObtain}}$ to obtain $\mathcal{A}$'s committed vector $\vec{m}_{|_{n_1}} = (m_1, \ldots, m_{n_1})$ (as well as the opening $\mathit{open}_{\vec{m}_{|_{n_1}}}$ of $\mathcal{A}$'s commitment V'. Unless $\mathcal{A}$ is able to break the soundness of the interactive proof system, $\vec{m}_{|_{n_1}}$ must be such that $m_1 = \mathsf{sk}_U$, where sk_U is the content of the perfectly binding commitment N. Then, $\mathcal{B}$ engages in an execution of $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ with its own challenger in the game of definition 8. This protocol results in $\mathcal{B}$ obtaining a P-signature σ on a vector $\vec{m}$ whose first n_1 coordinates are those of $\vec{m}_{|_{n_1}}$ (its remaining coordinates are attributes that are either chosen by $\mathcal{A}$ and revealed to the organization or chosen by the latter). Then, by executing the simulator $\mathsf{SimIssue}(V', (m_{n_1+1}, \ldots, m_n), \sigma)$ (which exists thanks to the signer privacy property of the P-signature), $\mathcal{B}$ simulates an execution of $\mathsf{SigIssue}$ (in interaction with $\mathcal{A}$ who runs $\mathsf{SigObtain}$) the outcome of which is $\mathcal{A}$ obtaining the P-signature σ as a credential.

The game ends with $\mathcal{A}$ outputting a pseudonym N_2, an index $i \in [0, n]$, a relation $R \in \mathcal{R}_1 \cup \mathcal{R}_2$, a vector $\vec{X} \in \mathcal{D}^n$ and a proof π_C. At this step, $\mathcal{B}$ parses π_C as $(C_{x_1}, C_{W_1}, C_\sigma, \pi_1, C_{W_1}, C'_\sigma, \pi_2, \pi_3)$ and flips a fair coin $\beta \overset{R}{\leftarrow} \{0, 1\}$. We note that, by the perfect soundness of the non-interactive proof π_3, C_{x_1} is guaranteed to contain the same value sk_U as the pseudonym N_2. If $\beta = 0$, the reduction $\mathcal{B}$ outputs $(C_\sigma, C_{W_1}, \pi_1, \mathsf{claim})$ where $\mathsf{claim} = (R^{\mathsf{EQ}}, 1, C_{x_1})$. If $\beta = 1$, it rather outputs $(C'_\sigma, C_W, \pi_2, \mathsf{claim})$ where $\mathsf{claim} = (R, i, \vec{X})$. We argue that, if $\mathcal{A}$ has advantage ε, $\mathcal{B}$ has advantage $\varepsilon/2$. To see why, we distinguish two situations:

1. $\mathsf{Extract}(C_\sigma) = \mathsf{Extract}(C'_\sigma)$ does not correspond to any σ that $\mathcal{B}$ obtained by running $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ in interaction with its own challenger.
2. $\mathsf{Extract}(C_\sigma) = \mathsf{Extract}(C'_\sigma)$ corresponds to σ_j, which stands for the j-th signature that $\mathcal{B}$ obtains from its challenger in the game of definition 8.

In case 1, it is easy to see that $\mathcal{B}$ always breaks the security of the P-signature scheme, no matter what the value of β is. Therefore, we only need to worry about situation 2. By assumption, we know that the j-th vector $\vec{m}_j$ that $\mathcal{B}$ used when running $\mathsf{SigObtain}$ was such that either $m_{j,1} \neq \mathsf{sk}_U$, where $\mathsf{sk}_U = F^{-1}(\mathsf{Extract}(C_{x_1}))$, or $R(i, \vec{m}_j, \vec{X}) = 0$. If $\beta = 0$, $\mathcal{B}$ is thus betting on the former case whereas, if $\beta = 1$, it hopes for the situation $R(i, \vec{m}_j, \vec{X}) = 0$. Since $\beta \in_R \{0, 1\}$ is chosen independently of $\mathcal{A}$'s view, the announced result follows. □

The proof of anonymity proceeds exactly like the proof of anonymity in [3].

Theorem 5. *The scheme provides anonymity if the interactive proof system used in* $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$ *is zero-knowledge and if the P-signature is zero-knowledge.*

Proof. The proof uses a sequence of hybrid experiments $H_0, H_1, \ldots, H_5$ where H_0 is the real experiment and H_5 is an experiment where the adversary only sees proofs that are simulated (using the simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimProve}_1, \mathsf{SimProve}_2, \mathsf{SimEqComProve})$ of the P-signature) without using any witness.

H_0: is the real experiment. Namely, the adversary interacts with real users and organizations running the protocol. In addition, the adversary is allowed to act as a dishonest organization and run $\mathsf{CredIssue}$ during executions of $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$ with honest users.

H_1: is like the original experiment H_0 but the adversary is given simulated public parameters params_s produced by $(\mathsf{params}_s, \tau) \leftarrow \mathsf{SimSetup}(\lambda)$. Everything else proceeds as in H_0 and, in particular, the adversary still observes real proofs generated by $\mathsf{SigProve}_1$, $\mathsf{SigProve}_2$ and $\mathsf{EqComProve}$. The simulation trapdoor τ is not used here. By the zero-knowledge property of the P-signature, this change is not noticeable by the adversary.

H_2: is identical to H_1 with one difference. Namely, when honest users interact with organizations in the registration of their pseudonyms, they still honestly generate their pseudonyms. However, they produce the zero-knowledge proof of knowledge of $(\mathsf{sk}_U, open_{\mathsf{sk}_U})$ using the ZK simulator instead of the real witnesses. The zero-knowledge property of the interactive proof system guarantees that the adversary's view is not affected by this change.

H_3: is as H_2 with the difference that, at each invocation of $\mathsf{EqComProve}$ in the computation of π_3 and π_4 as part of $\mathsf{CredProve}$, the adversary is provided with simulated proofs generated by $\mathsf{SimEqComProve}$. In particular, instead of generating commitments to (ρ_1, ρ_2, ρ_3) using the state information transmitted by $\mathsf{SigProve}_1$ and $\mathsf{SigProve}_2$ at each run of $\mathsf{CredProve}$ as $\mathsf{EqComProve}$ did, $\mathsf{SimEqComProve}$ rather uses commitments to 0 (which does not change anything for $\mathcal{A}$ as those commitments are perfectly hiding). Since the P-signature is zero-knowledge, this change can be made without the adversary noticing.

H_4: in this experiment, all proofs produced by $\mathsf{SigProve}_1$ and $\mathsf{SigProve}_2$ are replaced by simulated proofs obtained from $\mathsf{SimSigProve}_1$ and $\mathsf{SimSigProve}_2$ that produce NIZK proofs without even knowing P-signatures σ (as established by theorem 3). Hence, by the ZK property of the P-signature, we know that this will not make any difference w.r.t. H_3 in $\mathcal{A}$'s view.

H_5: is like H_4 but, at each run of CredObtain, simulated honest users run algorithm SimObtain (which exists due to the user privacy property of the P-signature) instead of the real algorithm SigObtain. This change is allowed since, from experiment H_4 onwards, users to not use their credentials to generate proofs any longer. The user privacy property of the P-signature ensures that $\mathcal{A}$'s view is not noticeably altered by this modification.

H_6: is identical to H_5 but, when registering their pseudonyms, users always compute them as commitments to $1_{\mathbb{G}}$. This change is clearly conceptual as those commitments have the same distribution either way.

It is easy to observe that, in experiment H_6, the adversary is interfacing with a full-fledged simulator that emulates the behavior of real users without using any secret. □

E.3 Efficiency

Our block-wise P-signature scheme is fairly efficient for such a primitive: each proof consists of about hundred group elements (at most 54784 bits for a 128-bit security level on supersingular curves) and their verification always requires less than 20 pairing evaluations and multi-exponentiations with small exponents.

The most expensive part of their generation is the witness generation: the algorithm Witness-Gen requires one multi-exponentiation using n base elements. The computational cost of a naive implementation (without pre-computation) is similar to the cost n single-base exponentiations but, using classical techniques, it can be reduced to the cost of one exponentiation (with a pre-computation stage and a storage of 2^n elements in $\mathbb{G}$) and time-memory tradeoffs are possible: witnesses can be generated using t exponentiations at the expense of storing $2^{n/t}$ pre-computed elements.

Table 1 summarizes the costs of the P-signature generation (after the generation of the witness) and verification. The multi-exponentiations mentioned in the P-signature generation are for constant-size basis (*i.e.*, independent of n).

	Size (# of elements in $\mathbb{G}$)	Generation	Verification [7]
Algorithm $\mathsf{SigProve}_1$			
$R = R^{\mathsf{EQ}}$	80	$80 \cdot$ MultiExp	$18 \cdot$ Pairing
$R = R^{\neg\mathsf{EQ}}$	101	$101 \cdot$ MultiExp	$20 \cdot$ Pairing
Algorithm $\mathsf{SigProve}_2$			
$R = R^{\mathsf{IP}}$	65	$66 \cdot$ MultiExp	$15 \cdot$ Pairing
$R = R^{\neg\mathsf{IP}}$	104	$105 \cdot$ MultiExp	$20 \cdot$ Pairing
$R = R^{\mathsf{EQ}}$	77	$77 \cdot$ MultiExp	$16 \cdot$ Pairing
$R = R^{\neg\mathsf{EQ}}$	107	$107 \cdot$ MultiExp	$20 \cdot$ Pairing

Table 1. Efficiency of the P-signature protocol

For instance, a proof generated by $\mathsf{SigProve}_1$ for the relation $R = R^{\mathsf{EQ}}$ consists in the commitment of the 12 elements σ_j (for $j \in \{1, \ldots, 7\}$), θ_1, W and $X_{i,1}, X_{i,2}, X_{i,3}$ for some $i \in \{1, \ldots, n\}$ and proof elements that they satisfy the relations (8)-(12) (*i.e.*, three (general) pairing product equations, five linear pairing product equations and one linear multi-exponentiation). Such a proof is therefore made of $12 \times 9 + 3 \times 9 + 5 \times 3 + 1 \times 2 = 80$ elements from $\mathbb{G}$ and its verification requires the computation of 18 pairings[4] and multi-exponentiations in $\mathbb{G}$ (using the batching techniques from [7]). Table 1 summarizes the computational cost of our P-signature scheme.

From a bandwidth point of view, the efficiency of our anonymous credential system is comparable with that of previous non-interactive constructions like [3]: in the worst case, a credential generated by $\mathsf{CredProve}$ consists in 134 group elements in $\mathbb{G}$ (68608 bits for a 128-bit security level). From a computational standpoint, the number of (dominant) pairing operations in $\mathsf{CredVerify}$ is always less than 24 (if we remove redundant group elements and further batch the verification equations). In comparison with the non-interactive scheme of [3], the main overhead is the number of exponentiations (which is linear in n) when it comes to generate proofs.

E.4 Predicates Handled using Inner Products

In [38], Katz, Sahai and Waters showed how to express a number of predicates using inner products. For example, consider a polynomial $P[Z] = \rho_0 + \rho_1 Z + \cdots + \rho_{n-1} Z^{n-1} \in \mathbb{Z}_p[Z]$ and, for some $w \in \mathbb{Z}_p$, define the relation $R^{\mathsf{poly}}(P, w) = 1$ iff $P(w) = 0$. This relation can be expressed by encoding the coefficients of $P[Z]$ as $\vec{m} = (\rho_0, \ldots, \rho_{n-1})$ and defining the vector $\vec{X} = (1, w, w^2, \ldots, w^{n-1})$ in such a way that $P(w) = 0 \Leftrightarrow \vec{m} \cdot \vec{X} = 0$. Multivariate polynomials can be handled in a similar way, as summarized in the table hereafter.

We consider vectors $\vec{m} = (m_1, \ldots, m_n)$ that contain the user's private key in their first coordinate (*i.e.*, $m_1 = \mathsf{sk}$) and that other coordinates encode the user's attributes. To prove statements about their attribute set without revealing their key sk, users can simply prove that $\vec{m}$ is orthogonal to $\vec{X} = (0, x_1, \ldots, x_{n-1})$, for the appropriate choice of $(x_1, \ldots, x_{n-1})$.

Proving the inclusion of given attributes. Suppose that users' attributes are encoded in an unordered set S of size $|S| \leq n-2$. If users want to prove that a specific attribute $x \in \mathbb{Z}_p$ is in their set S, they can define $\vec{m} = (\mathsf{sk}, \rho_0, \ldots, \rho_{n-2})$ using the coefficients of $P[Z] = \prod_{\omega \in S}(Z - \omega) = \sum_{k=0}^{n-2} \rho_k Z^k$ and merely prove that $\vec{m} \cdot \vec{X} = 0$ where $\vec{X} = (0, 1, x, \ldots, x^{n-2})$.

Proving OR statements about given attributes. By exchanging the roles of $\vec{X}$ and $\vec{m}$ in the above encoding, we obtain a simple way for users to prove that one of their attributes is contained (or not) in a list of public attributes. Suppose that users have at most $t = |S|$ attributes and that d is a pre-specified bound on the size of public lists where users are allowed to hide their attributes. If the scheme is set up for $n = t \cdot (d+1) + 1$, the vector $\vec{m} = (\vec{\omega}_1 | \ldots | \vec{\omega}_t | sk)$ must be a concatenations of t sub-vectors of the form $\vec{\omega}_\ell = (1, \omega_\ell, \omega_\ell^2, \ldots, \omega_\ell^d)$ for each $\ell \in [1, t]$. In order to prove that attribute $\omega_\ell \in S$ is contained within $S_{pub} = \{s_1, \ldots, s_d\}$, for some $\ell \in [1, t]$, users can define $\vec{X} = \big(0, \mathbf{0}^{(\ell-1)\cdot(d+1)} | (\rho_0, \ldots, \rho_d) | \mathbf{0}^{n-\ell\cdot(d+1)}\big)$ using the coefficients of the polynomial $P[Z] = \prod_{k=1}^{d}(Z - s_k) = \rho_0 + \rho_1 Z + \cdots + \rho_d Z^d$ and then prove that $\vec{m} \cdot \vec{X} = 0$. This entails that, when executing the $\mathsf{CredIssue} \rightleftarrows \mathsf{CredObtain}$ protocol, users have to prove that sub-vectors $\{\vec{\omega}_\ell\}_{\ell=1}^{t}$ are all of the form $\vec{\omega}_\ell = (1, \omega_\ell, \omega_\ell^2, \ldots, \omega_\ell^d)$. Fortunately, standard Σ-protocols allow to do this with $O(n)$ exponentiations using Horner's polynomial evaluation algorithm. The asymptotic communication complexity is not affected as $O(n)$ elements still have to be transmitted as in the basic proof of knowledge of an opening of $\sigma_6 = \mathrm{VecCom}(m_1, \ldots, m_n; r)$.

By combining the above technique with our proof for the non-zero evaluation of inner products, we obtain a very efficient protocol allowing users to prove that they are above 21 years old (by setting $d = 21$) for example.

[4] A naive verification procedure requires 252 pairing evaluations.

Proving non-exclusive OR statements. In situations where users' attributes sets consist of ordered vectors $(M_1, \ldots, M_t)$, it was shown in [38] that inner products can handle disjunctions of the form $(M_1 = X_1) \vee \ldots \vee (M_t = X_t)$ when t is not too large (since vectors of size $n = 2^t$ are needed). With $t = 3$, for example, the encoding $\vec{m} = (\mathsf{sk}, M_1M_2M_3, M_1M_3, M_2M_3, M_1M_2, M_1, M_2, M_3, 1)$, $\vec{X} = (0, 1, -X_2, -X_1, -X_3, X_2X_3, X_1X_3, X_1X_2, -X_1X_2X_3)$ guarantees that $\vec{m} \cdot \vec{X} = 0$ if and only if $(M_1 - X_1)(M_2 - X_2)(M_3 - X_3) = 0$. Of course, some extra work is needed to prove the well-formedness of the committed part of $\vec{m}$ during the execution of $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$.

Proving AND statements using limited interaction. When $\vec{m} = (\mathsf{sk}, M_1, \ldots, M_{n-1})$ is an ordered attribute vector, a technique of [38] handles conjunctions $(M_1 = X_1) \wedge \ldots \wedge (M_{n-1} = X_{n-1})$ by proving that the polynomial $P[M_1, \ldots, M_{n-1}] = r_1(M_1 - X_1) + \cdots + r_{n-1}(M_{n-1} - X_{n-1})$, where $r_1, \ldots, r_{n-1} \in \mathbb{Z}_p$ are random coefficients, cancels in $(X_1, \ldots, X_{n-1})$. In our setting, the coefficients $\{r_j\}_{j=1}^{n-1}$ must be chosen by the verifier (as it is the only way to guarantee their uniform distribution and their independence of public values $(X_1, \ldots, X_{n-1})$). Using these random coefficients, the prover can provide evidence that the vector $\vec{m} \odot (0, X_1, \ldots, X_{n-1})^{-1}$ is orthogonal to $(0, r_1, \ldots, r_{n-1})$ by generating a proof for the equation $e\big(\sigma_6 \cdot \prod_{j=2}^{n} g_{n+1-j}^{X_{j-1}}, \prod_{j=2}^{n} g_j^{r_{j-1}}\big) = e(g, W)$.

As it turns out, we need two rounds of interaction here: when the prover decides to prove that $(M_1 = X_1) \wedge \ldots \wedge (M_{n-1} = X_{n-1})$, a challenge $\{r_j\}_{j=1}^{n-1}$ has to be sent by the verifier to randomize the polynomial $P[M_1, \ldots, M_{n-1}]$. Fortunately, the amount of interaction can be minimized (in particular, the communication complexity can remain independent of n) by having the verifier send a single challenge value $r \in \mathbb{Z}_p$ which allows defining $r_j = r^{j-1}$ for $j = 1$ to $n - 1$. Thanks to the Schwartz-Zippel lemma [46], the prover can only cheat if $r \in_R \mathbb{Z}_p$ is a root of the polynomial $P[Z] = (M_1 - X_1) + Z(M_2 - X_2) + \cdots + Z^{n-2}(M_{n-1} - X_{n-1})$, which occurs with negligible probability. The same technique allows dealing with negated conjunctions $(M_1 \neq X_1) \vee \ldots \vee (M_{n-1} \neq X_{n-1})$ by proving the non-orthogonality of $\vec{m} \odot (0, X_1, \ldots, X_{n-1})^{-1}$ and $(0, 1, r, r^2, \ldots, r^{n-2})$.

Batch proofs of inclusion using limited interaction. The Schwartz-Zippel lemma can also be used to simultaneously prove the inclusion of several attributes when users' attributes are encoded as the root of a polynomial $P[Z] = \prod_{\omega \in S}(Z - \omega)$ whose coefficients are in $\vec{m}$. Namely, suppose that the prover wishes to convince the verifier that $x_1, \ldots, x_t \in S$, where S denotes his attribute set. To this end, he defines $\vec{X}_1 = (0, 1, x_1, \ldots, x_1^{n-2})$, $\ldots$, $\vec{X}_t = (0, 1, x_t, \ldots, x_t^{n-2})$ and uses them to construct $\vec{Y} = \vec{X}_1 + r \cdot \vec{X}_2 + \cdots + r^{t-1} \cdot \vec{X}^t$, where $r \in_R \mathbb{Z}_p$ is a random challenge sent by the verifier, and prove that $\vec{m} \cdot \vec{Y} = 0$. This convinces the verifier that $\vec{m} \cdot \vec{X}_j = 0$ for each $j \in [1, t]$ with overwhelming probability. Indeed, if there exists $j \in [1, t]$ such that $\vec{m} \cdot \vec{X}_j \neq 0$, we can only have $\vec{m} \cdot \vec{Y} = 0$ if the verifier accidentally chooses r as a root of a specific polynomial of degree at most $t - 1$ and this occurs with probability smaller than t/p, which is negligible.

CNF and DNF Formulas. Similarly to [38], CNF and DNF formulas can be expressed by combining the above techniques. For example, the 3-CNF

$$(M_1 = X_1) \vee (M_2 = X_2) \vee (M_3 = X_3) \wedge (M_2 = X_2) \vee (M_4 = X_4) \vee (M_5 = X_5)$$

requires to encode the polynomial $(M_1 - X_1)(M_2 - X_2)(M_3 - X_3) + r(M_2 - X_2)(M_4 - X_4)(M_5 - X_5)$, for some random $r \in \mathbb{Z}_p$, using vectors of length $n = 16$. In general, a t-CNF with k clauses can be expressed with vectors of size $n = k \cdot 2^t$.

The table below summarizes the basic predicates that easily translate in terms of inner products.

Predicate	Implementation as a polynomial
$(z = I_1) \vee (z = I_2) \cdots \vee (z = I_{n-1})$	$f_{\mathsf{OR},I_1,I_2,\ldots,I_{n-1}}(z) = \prod_{j=1}^{n-1}(z - I_j) = 0$
$(z_1 = I_1) \vee (z_2 = I_2) \vee \cdots \vee (z_{n-1} = I_{n-1})$	$f_{\overline{\mathsf{OR}},I_1,I_2,\ldots,I_{n-1}}(z_1,\ldots,z_{n-1}) = \prod_{j=1}^{n-1}(z_j - I_j) = 0$
$(z_1 = I_1) \wedge (z_2 = I_2) \cdots \wedge (z_{n-1} = I_{n-1})$	$f_{\mathsf{AND},I_1,I_2,\ldots,I_{n-1}}(z_1,\ldots,z_{n-1}) = \sum_{j=1}^{n-1} r_j(z_j - I_j) = 0$
$(z_1 \neq I_1) \vee (z_2 \neq I_2) \cdots \vee (z_{n-1} \neq I_{n-1})$	$f_{\mathsf{OR\text{-}NOT},I_1,I_2,\ldots,I_{n-1}}(z_1,\ldots,z_{n-1}) = \sum_{j=1}^{n-1} r_j(z_j - I_j) \neq 0$
$(z \neq I_1) \wedge (z \neq I_2) \wedge \cdots \wedge (z_{n-1} \neq I_{n-1})$	$f_{\mathsf{AND\text{-}NOT},I_1,I_2,\ldots,I_{n-1}}(z) = \prod_{j=1}^{n-1}(z - I_j) \neq 0$
$(z_1 \neq I_1) \wedge (z_2 \neq I_2) \wedge \cdots \wedge (z_{n-1} \neq I_{n-1})$	$f_{\overline{\mathsf{AND\text{-}NOT}},I_1,I_2,\ldots,I_{n-1}}(z_1,\ldots,z_{n-1}) = \prod_{j=1}^{n-1}(z_j - I_j) \neq 0$

Proving exact threshold statements over binary attributes. When users' attributes are binary, another technique – also suggested in [38] – allows a user to convince a verifier that he holds *exactly* t attributes appearing in a list of ℓ public binary attributes. Let $\mathcal{U}$ be an attribute universe of size $u = |\mathcal{U}|$. If $S \subset \mathcal{U}$ is the subset of attributes held by the prover and $X \subset \mathcal{U}$ is the ℓ-set of public attributes, S is encoded as a vector $\vec{m} = (\mathsf{sk}, 1, \tilde{M}_1, \ldots, \tilde{M}_u) \in \{0,1\}^{u+1} \times \mathbb{Z}_p$ of length $n = u + 2$, where $\tilde{M}_i = 1$ if attribute i is in S and $\tilde{M}_i = 0$ otherwise. The set $X \subset \mathcal{U}$ is translated to the vector $\vec{X} = (0, -t, \tilde{x}_1, \ldots, \tilde{x}_u) \in \mathbb{Z}_p \times \{1, \ldots, \ell\} \times \{0,1\}^u$ where $\tilde{x}_j = 1$ if and only if attribute j belongs to X. Proving the statement then amounts to demonstrating that $\vec{m} \cdot \vec{X} = 0$. Again, the encoding of $\vec{m}$ requires the user to prove that each committed $\tilde{M}_i$ is a binary value during the execution $\mathsf{CredIssue} \leftrightarrows \mathsf{CredObtain}$ but this can be achieved using standard techniques.

In [25], a very similar technique was suggested to prove that binary sub-vectors $(\tilde{M}_1, \ldots, \tilde{M}_u)$ and $(\tilde{x}_1, \ldots, \tilde{x}_u)$ are exactly $t \in [1, u]$ elements apart in terms of Hamming distance. This technique was based upon the simple observation that this distance can be expressed as $\sum_{j=1}^{u} \tilde{M}_j(1 - 2\tilde{x}_j) + \sum_{j=1}^{u} \tilde{x}_j$ which, in our setting, easily translates in terms of inner products using the encoding $\vec{m} = (\mathsf{sk}, 1, \tilde{M}_1, \ldots, \tilde{M}_u)$ and $\vec{X} = (0, -t + \sum_{j=1}^{u} \tilde{x}_j, 1 - 2\tilde{x}_1, \ldots, 1 - 2\tilde{x}_u)$.

Proving inexact thresholds. The above techniques only allow proving *exact* threshold and Hamming distance predicates and it would be interesting to extend them so as to prove statements of the form "sets S and X have at most d attributes in common" or "sets S and X are at most d attributes apart". One way to solve this problem is to have the prover only publicize the value t in committed form and provide evidence that t belongs to some small verifiable range $[1, d]$ with $d \in [1, u]$. The techniques of Camenisch, Chaabouni and shelat [13] provide a solution to this problem if we increase the signer's public key and let it consist of $O(n^2)$ elements.

We illustrate this for inexact threshold statements (namely "S and X have at most d common attributes") for which it is convenient to define $\vec{X} = (x_1, x_2, \ldots, x_n) = (0, -t, \tilde{x}_1, \ldots, \tilde{x}_u)$. Instead of proving equation (15) for a fully public vector $\vec{X}$, the user does it in the case where x_2 is committed and $(x_2, \ldots, x_n)$ are public. Namely, the prover computes and discloses a Groth-Sahai commitment $\vec{C}_t = \mathsf{GSCom}'(t, open_t)$ to $t \in \mathbb{Z}_p$ and proves that the committed t satisfies

$$e\big(\sigma_6, g_2^{-t} \cdot \prod_{j=1}^{u} g_{j+2}^{\tilde{x}_j}\big) = e(g, W), \tag{34}$$

using the committed witness $W = \prod_{j=1}^{u+2} W_j^{x_j}$, where $W_j = \prod_{k=1, k \neq j}^{n} g_{n+1-k+j}^{x_j}$. To complete the proof, the only problem left is to provide evidence that the value hidden by C_t falls in the range $[1, d]$ (since the latter is small, this extractability of t from C_t is guaranteed and everything goes through in the proof of lemma 3 as explained by Remark 1 in appendix B) and this is where the technique of [13] comes into play. Instead of using a single key pair $(\mathsf{sk}, \mathsf{pk})$, the signer generates exactly $u = n - 2$ additional P-signature key pairs $(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_u, \mathsf{pk}_u)$ (note that all these P-signature instances only need to sign vectors containing a single message, so that they require a commitment key for an instance of (VecCom, VecOpen) where $n = 1$). For $j = 1$ to u, the secret key sk_j is used to generate a set of P-signatures $\Sigma_j = \{sig_{j,1}, \ldots, sig_{j,j}\}$ on integer values $\{1, \ldots, j\}$ and all signatures $\Sigma_1, \ldots, \Sigma_u$ are then set as part of the public key. Eventually, the prover has to merely prove his knowledge (using algorithm $\mathsf{SigProve}_1$) of a valid signature $sig_{d,t}$ on the message $t \in \mathbb{Z}_p$ contained in the commitments $\{\vec{C}_{T,j} = \mathsf{GSCom}(T_j, open_{T,j})\}_{j\in\{1,2,3\}}$, where $(T_1, T_2, T_3) = (g_1^t, g^t, g_{2n}^t)$, w.r.t. the public key pk_d. In addition, the prover uses $\mathsf{SigProve}_2$ to demonstrate his possession of a signature $\sigma = (\sigma_1, \ldots, \sigma_6, r)$ under pk as well as his knowledge of $(\Gamma, W) \in \mathbb{G}^2$ such that

$$e(\sigma_6, \Gamma \cdot \prod_{j=1}^{u} g_{j+2}^{\tilde{x}_j}) = e(g, W) \qquad\qquad e(T_2, g_2) \cdot e(g, \Gamma) = 1_{\mathbb{G}_T} \tag{35}$$

By doing so, the verifier will be convinced that the committed $t \in \mathbb{Z}_p$ is such that $t \in [1, d]$ and the same twist can be applied to prove inexact Hamming distance statements. We note that the P-signature of [3] was used in a similar way to construct non-interactive range proofs in [44].

F Generic Security of the n-FlexDHE Problem

Theorem 6. *Let $\mathcal{A}$ be an adversary in the generic group model that makes at most t group operation queries and pairing evaluations. On inputs $(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n}$ such that $g_i = g^{(\alpha^i)}$, for $i \in \{1, 2, \ldots, n, n+2, \ldots 2n\}$, and where $\alpha \xleftarrow{R} \mathbb{Z}_p^*$, the probability that $\mathcal{A}$ outputs a solution $(g^\mu, g_{n+1}^\mu, g_{2n}^\mu) \in (\mathbb{G}\backslash\{1_{\mathbb{G}}\})^3$, for some $\mu \in \mathbb{Z}_p^*$ for the n-FlexDHE Problem in symmetric bilinear groups is bounded by $O(t^2/p)$.*

Proof. We provide an analysis of the n-FlexDHE Problem in the (symmetric) generic bilinear group model [43]. $\mathcal{A}$ denotes an adversary against the n-FlexDHE Problem and $\mathcal{B}$ the simulator that emulates group operation using two lists of polynomials in $\mathbb{Z}_q[X]$: $L_1 = \{(F_{1,i}, \xi_{1,i}), i = 1, \cdots, t_1\}$ and $L_T = \{(F_{T,j}, \xi_{T,j}), j = 1, \cdots, t_T\}$ such that at step t, the following equality remains true $t_1 + t_T \leq t + 2n + 1$. The entries $\xi_{1,i}$ and $\xi_{T,j}$ are random strings used to represent elements in $\mathbb{G}$ and $\mathbb{G}_T$ respectively. The initialization step ($t = 0$) sets $t_1 = 2n, t_T = 0$ and defines a list L_1 of $2n+1$ polynomials $\{F_{1,1} = 1, F_{1,2} = X, \ldots, F_{1,n+1} = X^n, F_{1,n+3} = X^{n+2}, \ldots, F_{1,2n+1} = X^{2n}\}$ with corresponding random strings $\xi_{1,i}$ (that represents elements of the given input in $\mathbb{G}$). $\mathcal{A}$ is given as input the list L_1 of polynomials and may request either:

- **Multiplication of group elements in $\mathbb{G}$:** on input two representations $\xi_{1,i}$ and $\xi_{1,j}$. $\mathcal{B}$ increments t_1, computes the sum $F_{1,i}+F_{1,j}$ and sets $F_{1,t_1} = F_{1,i}+F_{1,j}$. If the resulting polynomial already appears in the list L_1 for some index $\ell < t_1$, then it sets $\xi_{1,t_1} \leftarrow \xi_{1,\ell}$, else it chooses a fresh random string for ξ_{1,t_1}. Note that group operations in $\mathbb{G}$ result in (at most t_1) univariate polynomials of degree at most $2n$.
- **Pairing computation:** Upon reception of a pairing operation request (consisting in two representations $\xi_{1,i}$ and $\xi_{1,j}$), $\mathcal{B}$ increments t_T, computes the product of the polynomial, $F_{1,i} \cdot F_{1,j}$ and sets $F_{T,t_T} = F_{1,i} \cdot F_{1,j}$. If the resulting polynomial already appears in the list L_T for some index $\ell \leq t_T$, then it $\xi_{T,t_T} \leftarrow \xi_{T,\ell}$, else it chooses a fresh random string for ξ_{T,t_T}. Note that pairing operations in $\mathbb{G}_T$ result in polynomials of degree at most $4n^2$.
- **Multiplication of group element in $\mathbb{G}_T$:** $\mathcal{B}$ proceeds exactly as before except that group operations are answered using L_T and result in polynomials of degree at most $4n^2$.

We will evaluate later the probability that the simulation above deviates from real oracle responses. At the end, $\mathcal{A}$ outputs a triple $(\xi_{1,\ell_1}, \xi_{1,\ell_2}, \xi_{1,\ell_3})$ with $1 \leq \ell_i \leq t_1$ for $i \in \{1,2,3\}$.
Note that $\mathcal{A}$'s response is correct if the following equalities hold:

$$\begin{cases} F_{1,\ell_1} \cdot X^{n+1} - F_{1,\ell_2} = 0 \\ F_{1,\ell_1} \cdot X^{2n} - F_{1,\ell_3} = 0 \end{cases}$$

By degree considerations, it is easy to see that $\mathcal{A}$ cannot produce such a valid triple. Thus, the winning probability of $\mathcal{A}$ is bounded by the probability that it detects deviation of the simulated oracle from the real one. To quantify this, $\mathcal{B}$ chooses a uniformly and independently random assignment for X in $\mathbb{Z}_p$. If the evaluations of two polynomials are equal, our simulation would output two different representations whereas a real oracle would output the same representation. Since polynomials are of degree at most $2n$ in two variables, after t queries, using the Schwartz-Zippel lemma [46], the probability that $\mathcal{A}$ detects the simulation is bounded by $2n(t+2n+1)^2/p$.
$\square$