Anonymous Credentials

1 Abstract

The digital identity landscape is on the cusp of massive adoption. eiDAS-2 is the EU's framework for digital identity, which will make it mandatory for 400 million EU citizens to own interoperable identity wallets containing legal credentials issued by their national governments by September 2025. In this paper, we identify 2 functionalities missing in current literature, we implement, to the best of our knowledge, the most efficient and feature-rich privacy-preserving identity system with a full suite of mandatory functionality expected by governments and large organizations in current non-private identity systems. We are the first identity system to support privately linked "context/pairwise" credentials (separate credentials and not pseudonyms); we compare the functionality and concrete efficiency differences between the two candidate signature schemes, BBS+ and PS, used for Anonymous Credentials, and their ability to support multi-show, threshold, blind issuance, and anonymous authentication. Our identity system supports accountable privacy for Sybil-resistance, Revocation, Key Recovery, Credential Expiration. Finally, we implement everything in rust using the Arkworks Library and show that contrary to popular belief and the currently available benchmarks, standard PS signatures are efficient, furthermore, we reduce the heavy pairing computation by leveraging the cyclotomic subgroup to compute all pairings in their intermediate representation and performing a single final exponentiation and show that it reduces computation by x percent, a trick we haven't seen elsewhere.

2 Intro

Contributions We present a comprehensive privacy-preserving decentralized identity system that improves upon the state of the art, offering accountable privacy, complex identity support, decentralization, and efficient implementation.

- **Feature-Rich Implementation:** We implement a full suite of mandatory functionality expected in non-private identity systems, including Sybil-resistance, revocation, key recovery, and credential expiration.
- **Novel Functionality:** We introduce privately linked "context/pairwise" credentials, a feature missing in current literature.
- **Signature Scheme Comparison:** We provide a comparative analysis of BBS+ and PS signature schemes for Anonymous Credentials, evaluating their efficiency and support for multi-show, threshold, blind issuance, and anonymous authentication.
- **Efficient Implementation:** We implement the system in Rust using the Arkworks Library, demonstrating practical efficiency of standard PS signatures (contrary to theoretical analysis and prior benchmarks analysis) and introducing an optimization technique for pairing computations.
- **Accountable Privacy:** We maintain privacy while supporting accountability features required by governments and large organizations.

Insufficiencies of prior approaches

- CanDID is the first paper to address the linked context/pairwise credentials; however, do so in a non-private manner which we improve on. CanDID supports sybil-resistance, revocation, and key recovery which we also support. CanDID supports legacy compatibility and private sanctions screening which was designed to solve prior problems which are no longer required as now governments will be issuing DID's widescale. We improve their privacy notions and solve their open problem by implementing anonymous credentials

- Hades is a state-of-the-art DID system optimized for blockchain using zkSNARKS, and supporting pseudonyms, fine-grained sybil-resistance, and audit capability. We satisfy the same functionality and improve on their efficiency by employing Σ protocols rather than zkSNARKS.
-

3 Preliminaries and Assumptions

Notation A probabilistic polynomial time algorithm $\mathsf{Algorithm}(\mathsf{in}) \rightarrow \mathsf{out}$ receives an input in and returns an output out. $r \xleftarrow{\$} \mathbb{Z}_p$ r is sampled uniformly from the set of field elements modulo p, $h \leftarrow y$ is a deterministic assignment. $[n]$ denotes a sample space of $\{1, \ldots, n\}$. We assume type 3 bilinear pairings, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ over groups of prime order p, $g, \tilde{g}$ are uniformly chosen generators for $\mathbb{G}_1, \mathbb{G}_2$ such that $e(g, \tilde{g}) = g_t$. We use bold variables to denote vectors as $\mathbf{m} = [m_1, \ldots, m_\ell]$, $\mathbf{g} \in \mathbb{G}^\ell$, $\mathbf{x} \in \mathbb{Z}_p^\ell$, $\mathbf{g}^\mathbf{x} = \sum_{i=1}^{\ell} g_i^{x_i}$. We use multiplicative notation for $\mathbb{G}$ points i.e. $g^k = g \cdot g$ (k times)

Definition 1 (Negligible Function). *A function $\mu : \mathbb{N} \rightarrow \mathbb{R}$ is called negligible if for every positive polynomial $p(\cdot)$, there exists a value $N \in \mathbb{N}$ such that for all $n \geq N$ $\mu(n) < \frac{1}{p(n)}$*

Definition 2 (Bilinear map). *Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are multiplicative and $\mathbb{G}_T$ is multiplicative. Let g and h be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We call $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a bilinear map or pairing if it is efficiently computable and the following holds:*

Bilinearity: $e(g^a, \tilde{g^b}) = e(g, \tilde{g})^{ab} \quad \forall a, b \in \mathbb{Z}_p$.

Non-degeneracy: $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$, *i.e.,* $e(g, \tilde{g})$ *generates* $\mathbb{G}_T$.

If $\mathbb{G}_1 = \mathbb{G}_2$, then e is symmetric (Type-1) and asymmetric (Type-2 or 3) otherwise. For Type-2 pairings, there is an efficiently computable isomorphism $\Psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ but none from $\mathbb{G}_1 \rightarrow \mathbb{G}_2$; for Type-3 pairings, no efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ are known. Type-3 pairings are currently the optimal choice in terms of efficiency for a given security level.

4 System Model

Our identity system involves a user interacting with a registration authority, identity providers, and an auditor. We separate the concerns of the RA, IdP, and Auditor, but in practice, they could be one (threshold) entity of nodes or nodes of a blockchain.

- **User:** (U) a user holds a registration credential rcd and any number of context credentials ccd in their identity wallet. Their rcd contains a pid public identifier such as a passport number or email address, s their secret PRF key, which they use during context-credential generation, dsk the identity-based decryption key, and credential expiry exp. The user also holds context credentials ccd issued by Identity Providers IDP such as a driver's license, or a university bachelor's degree. To generate rcd, the user interacts with the RA, which involves a non-private interaction where U identifies themself with a previous login mechanism - this is a requirement for governments implementing Decentralized Identity. To generate ccd the user interacts
- **Registration Authority:** (RA) The RA runs an identity system outside this protocol's scope. It's used to store a mapping between U and their encrypted PRF key s for key escrow, used for accountability. On registration, U interacts with RA, RA learns the user's pid and verifies their rcd is sybil (based on the system) meanwhile they do not learn the prf key s. RA signs credentials with their keys rpk_{rcd}, rsk_{rcd} and Identity Based Encryption keys $\mathsf{rpk}_{ibe}, \mathsf{rsk}_{ibe}$.
- **Identity Provider:** (IDP) Identity providers issue and verify context credentials, they may run their own signature scheme or may use the RA.

Sam: change to credential provider

- **Auditor:** (AUD) The auditor runs a Threshold Encryption Scheme with keys (ask, apk) for key escrow. During issuance, a user encrypts s stored in the external identity system. When During audit or accountability, the Auditor receives the cipher text and decrypts it to handle accountability.

4.1 Objects

Registration Credential rcd U registers with RA and attests to their personal information, rcd contains

- pid personal identifier such as passport number or email address
- s their PRF key
- dsk their IBE decryption key
- exp their credential expiry date
- rcm a commitment to pid, s, dsk, exp
- σ a signature by RA over rcm issued by rpk_{rcd}, rsk_{rcd}. A credential that verifies and has details that are not revoked verifies the user is valid and attested to by the RA, e.g. a government body

Registration Authority's Information URI RA receives the following during rcd issuance. rcd contains $\mathsf{rcm} = CM.Com()$

- pid the user's personal identifier
- rcm the registration commitment used in the signature
- π the commitment opening proof
- TPKE.Enc(s)

Context Credential ccd

- contains a commitment ccm to to the following attributes
- s their PRF key
- ctx_{id} the context credential identifier
- CP_{id} the credential provider
- exp the credential expiration
- attrs specific user attributes from CP

Context Credential Information cci To create a new, linked, context credential, a U with rcd and valid precred can submit the following information for verification and issuance

- precred is a commitment to the context credential ccm signed by a CP
- π U rcd verifies
- nullif, vk, y, used for Sybil resistance. nullif is a nullifier based off the users PRF key and context credential id, in secret form. vk and y prove it's correctness.
-

0. user verifies their pre-credential 2. user verifies their rcd 3. user verifies equality of attributes between rcd 4. user proves opening of the commitment all artifacts are posted on chain for verification threshold committee signs over the commitment user now has a context credential signed by a threshold committee

4.2 Protocols

Create Registration Credential U interacts with RA the registration authority; in this trusted, non-cryptographic process, the user attests to their identity through a trusted login. U samples $\mathsf{s}_1 \leftarrow\!\$\ \mathbb{Z}_p$, generates a commitment $Com_1([0, \mathsf{s}_1, 0, 0], \alpha)$ and gives it to RA, RA samples $\mathsf{s}_2 \leftarrow\!\$\ \mathbb{Z}_p$ and generates $Com_2([\mathsf{pid}, \mathsf{s}_2, \mathsf{dsk}, \mathsf{exp}], 0)$, computes $Com_1 \cdot Com_2 =$ such that $\mathsf{s} = \mathsf{s}_1 \cdot \mathsf{s}_2$ and $\mathsf{rcm} = Com([\mathsf{pid}, \mathsf{s}, \mathsf{dsk}, \mathsf{exp}], \alpha)$. This interaction hides the PRF key s from RA but allows RA to be involved in its issuance to prevent replay attacks.
U runs IBE with pid their personal identifier
U computes their PRF key escrow with TPKE.Enc(s) and prove in zero-knowledge the ciphertext. Finally, RA signs rcm, returns the registration credential rcd and stores URI User Registration Information

Create Context Credential A user U with their registration credential rcd wants to be issued a new digital driver's license ccd. U first needs to prove to the credential provider CP they have a valid rcd issued by RA, second they must prove they satisfy a requirement on CP that connects their identity in rcd to their identity in CP to ensure CP issues the credential correctly. As this will differ for every CP, we do not go into detail; however, it may involve selective disclosure of information within rcd, or proving equality of attributes across user profiles. U generates a commitment $Com_1([\mathsf{s}, 0, 0], \alpha)$ and gives it to CP who generates $Com_2([0, \mathsf{ctx}_{id}, \mathsf{CP}_{id}], 0)$ and creates $Com_1 \cdot Com_2 = Com([\mathsf{s}, \mathsf{ctx}_{id}, \mathsf{CP}_{id}], \alpha)$. U proves the opening of the commitment and the equality of s between Com and their rcd. CP signs the commitment with their signing algorithm of choice as each CP will have different infrastructure. We call this a pre-credential $\mathsf{precred}$. A U with $\mathsf{precred}$ takes their artifacts to RA for credential issuance or posts artifacts on the blockchain for credential issuance.

5 Security Model

5.1 Adversarial Model

We use 2 different adversarial models, one to model internal threats from the threshold systems and one to model external threats such as malicious users, verifiers, or outside parties.

Internal Threats $\mathcal{A}_1$ can statically and actively corrupt up to t of n nodes for t $<$ n/3. We include attempts to forge signatures, compromise user identity during credential generation, link master and context credentials together.

External Threats $\mathcal{A}_2$ is used to model $EUF-CMA$ for our signature scheme, $IND-CCA1/2$ for encryption scheme, and Zero Knowledge Proofs.

Assumptions Identity verification is handled by a system outside of ours, for example, a current method used for verification. We have a trust assumption on the user's registration credential being generated honestly from this process.

5.2 System Goals

- **Accountable Privacy:** to simultaneously enable anonymous use of the system while retaining accountability, two paradoxical properties
- **Complex Identity Support:** enabling private pairwise connections between credentials to enable system owners to setup private, hierarchal ownership

- **Decentralized and Efficient:** components must support threshold cryptography and optimize for efficiency over simplicity as to benchmark against identity systems
- **Enhanced Identity:** Key recovery and user friendly addresses are a plus (tbc)

Syntax of Anonymous Identity System with Sybil Resistance and Revocation After public parameter creation, we assume each algorithm receives public parameters as input pp

- $\mathsf{Setup}(1^n) \xrightarrow{\$} \mathsf{pp}$: inputs the security parameter λ in unary, outputs system parameters pp

Sam: is this correct, or should I specify all the public parameters separately. I.e. Threshold public keys,

- $\mathsf{OrgKeygen}(1^n, n, t) \xrightarrow{\$} \{\mathsf{osk}, \mathsf{opk}, (\mathsf{osk}_1, \ldots, \mathsf{osk}_n)\}$: input the t of n threshold, output opk the public key, vk the verification key, and sk_i the shared secret key for each party.
- $\mathsf{UserKeygen}(1^n) \xrightarrow{\$} (\mathsf{mk}_u, \alpha)$ outputs mk_u the master key and α the user secret key.
- $\mathsf{ObtainMaster}(\mathsf{mcm}, \mathcal{UL}, \mathcal{RL}, \phi) \xrightarrow{\$} \mathsf{IssueMaster}$ user inputs their master commitment mcm, the user and revocation lists $\mathcal{UL}, \mathcal{RL}$ and ϕ the issuance statement. Outputs are sent to the org running $\mathsf{IssueMaster}$
- $\mathsf{IssueMaster}(\mathsf{mcm}', \pi_{\mathsf{rcm}}\pi_{\mathcal{UL}}, \pi_{\mathcal{RL}}, \phi) \xrightarrow{\$} (\sigma, \beta, \mathcal{UL}')$ the org receives the users randomized commitment mcm', the π_{mcm} the proof mcm satisfies ϕ and the proofs the user is Sybil and isn't revoked. Outputs $\perp$ and cancels if proofs aren't valid, else, outputs rcd the master credential, $\mathcal{UL}'$ the updated user list, β the encrypted user information for the audit committee.
- $\mathsf{ObtainContext}(\mathsf{ccm}, \mathsf{rcd}, \mathcal{UL}, \mathcal{RL}, \phi) \xrightarrow{\$} \mathsf{IssueMaster}$ user inputs their context commitment ccm, the master credential rcd, the user and revocation lists $\mathcal{UL}, \mathcal{RL}$ and ϕ the issuance statement for the context. User randomizes ccm, outputs are sent to the org running $\mathsf{IssueContext}$
- $\mathsf{IssueContext}(\mathsf{ccm}', \pi_{\mathsf{ccm}}, \pi_{\mathcal{UL}}, \pi_{\mathcal{RL}}, \phi)$ the org receives the users randomized context commitment ccm', π_{ccm} contains proof the context credential contains attribute derived from master cred and satisfies ϕ, $\pi_{\mathcal{UL}}, \pi_{\mathcal{RL}}$ prove the user is Sybil and isn't revoked. Outputs $\perp$ and cancels if proofs aren't valid, else, outputs ccd the context credential, $\mathcal{UL}'$ the updated user list.
- $\mathsf{Show}(\mathsf{cd}, \mathsf{cm}, \phi, \mathcal{RL}) \rightarrow \mathsf{Verify}$ show is run by the user, inputs their commitment cm and credential cm, ϕ the statement to prove and $\mathcal{RL}$ the revocation list, outputs sent to Verify.
- $\mathsf{Verify}(\mathsf{cd}', \mathsf{cm}'\pi_i, \phi, \mathcal{RL}) \rightarrow \{0, 1\}$ verify is run by a verifier and takes in π_i verification proofs that satisfy the statement ϕ and the proves the credential isn't $\in \mathcal{RL}$. Outputs 1 if successful or 0 if failure.
- $\mathsf{Revoke}(\mathcal{RL}, \beta) \rightarrow \mathcal{RL}'$ revoke is run by the accountability committee who requests the organisation decrypt β to find mk_u and add required pseudonyms to $\mathcal{RL}$. Returns $\mathcal{RL}'$ the updated revocation list

5.3 Ideal Functionality

$\mathcal{F}.\mathsf{IssueRegistrationCredential}(pid, s, Attr)$

5.4 Security Properties

Our anonymous credential system with Sybil resistance aims to achieve the following security and privacy properties:

1. **Sybil resistance**: An adversary cannot obtain more than one credential for the same context

2. **Unforgeability**: An adversary cannot forge credentials of honest users or use credentials belonging to other users
3. **PRF Key Privacy:** An adversary, or less than t malicious nodes, can't obtain the PRF key during credential issuance or key escrow
4. **Anonymity:** an adversary can't learn more than a user's public information during credential verification
5. **Blind Issuance:** an issuer can't learn about a user's private information during credential issuance
6. **Validity:** an adversary with a revoked or expired credential cannot verify successfully
7. **Unlinkability for Credential Verification:** verifiers can't collude and link different uses of the same credential
8. **Unlinkability for Identity Usage:** verifiers can't collude and link an individual with multiple context credentials through their different credential uses

Sam: Is the security of ZKP included in the above properties?

- Soundness: a malicious user can't create a valid proof for an incorrect relation between Γ and τ
- Zero-Knowledge: π reveals nothing beyond the validity of the relation

Definition 3 (Commitment scheme). *A commitment scheme is a tuple* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *of PPT algorithms where:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$ *takes security parameter* λ *(in unary) and generates the commitment key* ck*;*
- $\mathsf{Commit}_{\mathsf{ck}}(m) \to (C, r)$ *obtains commitment* C *from secret message* m *and an opening key* r *which may be the randomness used in the computation.*
- $\mathsf{Open}_{\mathsf{ck}}(C; m, r) \to b \in \{0, 1\}$ *verifies the opening of the commitment* C *to the message* m *provided with the opening hint* r*, outputting a decision as to whether* C *commits to* m*.*

Correctness holds if for all ck output by Setup and all messages m from message space M, Open $=$ 1 with probability 1 when $\mathsf{Commit}_{\mathsf{ck}}(m) \to (C, r)$ and $\mathsf{Open}_{\mathsf{ck}}(C; m, r) \to b \in \{0, 1\}$

Definition 4 (A commitment scheme is hiding and binding).

A commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ is hiding if for all PPT adversaries $\mathcal{A}$:

$$\left| \Pr\left[b_0 = b_1 : \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ (m_0, m_1, r) \leftarrow \mathcal{A}(\mathsf{ck}) \\ b \leftarrow \{0, 1\} \\ (C_b, r_b) \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m_b) \\ b' \leftarrow \mathcal{A}(\mathsf{ck}, r, C_b) \end{array} \right] - 1/2 \right| = \mathsf{negl}(n)$$

A commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ is binding if for all PPT adversaries $\mathcal{A}$:

$$\Pr\left[b_0 = b_1 \neq 0 \land m_0 \neq m_1 : \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C, m_0, m_1, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{ck}) \\ b_0 \leftarrow \mathsf{Open}_{\mathsf{ck}}(C, m_0, r_0) \\ b_1 \leftarrow \mathsf{Open}_{\mathsf{ck}}(C, m_1, r_1) \end{array} \right] \leq \mathsf{negl}(n)$$

Informally, C is binding if no adversary can open C with 2 different messages and opening key with greater than negligible probability.

Definition 5 (Vector Commitment scheme). *A vector commitment scheme is commitment scheme for a vector of messages denoted by* $\boldsymbol{m} = (m_1, \ldots, m_\ell) \in M$ *satisfying a position binding property:*

- $\mathsf{Open}_{\mathsf{ck}}(, C, \mathbf{m}, i, \pi) \to b \in \{0, 1\}$ *verifies the opening of the commitment* C *to the message vector* $\mathbf{m}$*, vector position* i*, and opening proof* π

$$\Pr\left[\begin{array}{l} 1 \leftarrow \mathsf{Open}_{\mathsf{ck}}(C, \mathbf{m}_0, i, \pi_0) \\ 1 \leftarrow \mathsf{Open}_{\mathsf{ck}}(C, \mathbf{m}_1, i, \pi_1) \\ m_0 \neq m_1 \end{array} \middle| \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C, \mathbf{m}, \mathbf{m}', i, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} \right] = \mathsf{negl}(n)$$

5.5 Construction

Pedersen commitment schemes are the basis of our signature/credential scheme.

Definition 6 (Rerandomizable Symmetric Pedersen Vector Commitment Scheme). *We instantiate the Pedersen Vector Commitment over both* $\mathbb{G}_1 and \mathbb{G}_2$ *for use within the PS signature. We denote* $\mathsf{cm} \in \mathbb{G}_1, \tilde{\mathsf{cm}} \in \mathbb{G}_2$ *and verify* $\mathsf{cm} \equiv \tilde{\mathsf{cm}}$ *by asserting* $e(\mathsf{cm}, \tilde{g}) = e(g, \tilde{\mathsf{cm}})$*. Let* $\mathbb{G}_1, \mathbb{G}_2$ *be cyclic groups of large prime order* p *with an efficient Type 3 pairing* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$*. For a vector of messages* $\mathbf{m} = (m_1, \ldots, m_\ell) \in \mathbb{F}_q^\ell$*, the rerandomizable Pedersen vector commitment scheme consists of the following PPT algorithms:*

– $\mathsf{CM.Setup}(1^\lambda, \ell, \boldsymbol{y}) \to \mathsf{ck}$: *compute* $\boldsymbol{y} \in \mathbb{Z}_p^\ell$, $\boldsymbol{g} \leftarrow g^{\boldsymbol{y}}$, $\mathsf{ck} \leftarrow (g, \boldsymbol{g})$

– $\mathsf{CM.Com}_{\mathsf{ck}}(\mathbf{m}, r) \to (\mathsf{cm}, \tilde{\mathsf{cm}})$: *parse* ck *as* $(g, \mathbf{g})$, *compute* $\mathsf{cm} \leftarrow \mathbf{g}^{\mathbf{m}} g^r, \tilde{\mathsf{cm}} \leftarrow \tilde{\mathbf{g}^{\mathbf{m}}} \tilde{g^r}$, *output* $(\mathsf{cm}, \tilde{\mathsf{cm}})$

– $\mathsf{CM.Rerand}_{\mathsf{ck}}(\mathsf{cm}, \tilde{\mathsf{cm}}, r') \to (\mathsf{cm}', \tilde{\mathsf{cm}'})$: *compute* $\mathsf{cm}' = \mathsf{cm} \cdot g^{r\,'}$, $\tilde{\mathsf{cm}'} = \tilde{\mathsf{cm}} \cdot g^{r\,'}$, *output* $\mathsf{cm}', \tilde{\mathsf{cm}'}$

The scheme satisfies the following properties:

– *Correctness: For all* ck *output by* Setup*, all message vectors* $\mathbf{m}$*, and all* $i \in \{1, \ldots, \ell\}$*, if* $(C, r) \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathbf{m})$*, then* $\mathsf{Open}_{\mathsf{ck}}(C, \mathbf{m}, i, (r, m_i)) = 1$ *with probability 1.*

– *Hiding: The commitment* C *reveals no information about the committed message vector* $\mathbf{m}$*.*

– *Binding: It is computationally infeasible to find two different message vectors* $\mathbf{m} \neq \mathbf{m}'$ *and openings* r, r' *such that* $\mathsf{Commit}_{\mathsf{ck}}(\mathbf{m}) = \mathsf{Commit}_{\mathsf{ck}}(\mathbf{m}')$*.*

– *Rerandomizability: For any commitment* C *with opening* r*,* $\mathsf{Rerand}(C, r)$ *produces a new commitment* C' *that is indistinguishable from a fresh commitment to the same message vector.*

5.6 Usage

1. Each credential attribute is an exponent of the commitment 2. Homomorphism

A Pedersen Commitment's algebraic structure enables additive homomorphism, given messages m_1, m_2 and randomness r_1, r_2 and commitments $\mathsf{C}_1(m_1[u]_1 + r_1[v]_1)$ and $\mathsf{C}_2(m_2[u]_1 + r_2[v]_1)$, $\mathsf{C}_1 \cdot C_2$ creates a new commitment $= \mathsf{C}((m_1 + m_2)[u] + (r_1 + r_2)[v])$. Using additive homomorphism, a Pedersen Commitment can be rerandomized:

– $\mathsf{COM.Rerand}(\mathsf{C}) \to ([\mathsf{C}']_1, t)$: generate rerandomizing secret $t \xleftarrow{\$} \mathbb{F}_q$, then $[\mathsf{C}']_1 := [\mathsf{C}]_1 + t[v]_1 = m[u]_1 + (t + r)[v]_1$

5.7 Signature Schemes

Sam: Update these, simplify, finish the final syntax for Rerandomizable Threshold Blind Signature

⟪ below is copied from a paper, slightly update needed ⟫ A digital signature scheme Sig is a set of PPT algorithms $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$:

– $\mathsf{Sig.Setup}(1^{\mathsf{n}}) \xrightarrow{\$} \mathsf{pp}$: Setup takes a security parameter λ as input and outputs public parameters p and a message space $\mathcal{M}$

– $\mathsf{Sig.Keygen}(\mathsf{pp}) \xrightarrow{\$} (\mathsf{sk}, \mathsf{pk})$: Key Generation takes the system parameters as input and outputs a secret key sk and public key pk

– $\mathsf{Sig.Sign}(\mathsf{sk}, m) \xrightarrow{\$} (\sigma)$: Signing algorithm takes as input the secret key sk and message $m \in \mathcal{M}$ and outputs a signature σ.

– $\mathsf{Sig.Verify}(\mathsf{pk}, m, \sigma) \to 1/0$: Verify takes as input the public key pk, the message m and signature σ and outputs 1 for acceptance or 0 for rejection

Rerandomizable PS Signatures We use the Pointcheval Sanders signature as the base of the protocol due to privacy preserving algebraic properties, it's efficiency advantages over CL and BBS+, its versatility and ability to support threshold. Key properties are the signature components are 2 G1 components, the public key is in G2, it supports signature rerandomization

PS Signature

- $\mathsf{PS.KeyGen}(1^\lambda, \mathsf{ck}) \to (\mathsf{sk}, \mathsf{vk})$: select $x \leftarrow\$ \mathbb{Z}_p$, set $(\mathsf{sk}, \mathsf{vk}) \leftarrow (g^x, \tilde{g}^x)$
- $\mathsf{PS.Sign}_{\mathsf{ck}}(\mathsf{sk}, \mathsf{cm}, u) \to \sigma$:

  Sam: run PoK of cm with respect to ck CM.Open or ZKPoK?

  Signer inputs u, parses ck as $(g, \mathbf{g}, \tilde{g}, \tilde{\mathbf{g}})$. Computes $\sigma_1 \leftarrow g^u, \sigma_2 \leftarrow (\mathsf{sk} \cdot \mathsf{cm})^u$. Outputs $\sigma \leftarrow (\sigma_1, \sigma_2)$
- $\mathsf{PS.Verify}_{\mathsf{ck}}(\mathsf{vk}, \mathsf{cm}, \tilde{\mathsf{cm}}, \sigma) \to \{0,1\}$: parse ck as $(g, \mathbf{g}, \tilde{g}, \tilde{\mathbf{g}})$ and σ as σ_1, σ_2. Assert $e(\mathsf{cm}, \tilde{g}) = e(g, \tilde{\mathsf{cm}})$ and $e(\sigma_2, \tilde{g}) = e(\sigma_1, \mathsf{vk} \cdot \tilde{\mathsf{cm}})$.
- $\mathsf{PS.Rerand}(\sigma, r_\Delta, u_\Delta) \to \sigma'$. Parse σ as (σ_1, σ_2). Compute $\sigma_1' \leftarrow \sigma_1^{u_\Delta}$, $\sigma_2' \leftarrow (\sigma_2 \cdot \sigma_1^{r_\Delta})^{u_\Delta}$, output $\sigma' \leftarrow (\sigma_1', \sigma_2')$

Threshold PS Signature

Secret Sharing A t of n secret sharing scheme, (t, n), shares secret x amongst n nodes. x can be reconstructed with t shares, fewer shares reveal nothing about x.

Definition 7 (Secret Sharing). *A (t, n) secret sharing scheme* SS *is a tuple of* PPT *algorithms* $(\mathsf{Share}, \mathsf{Combine})$ *over message space $x \in X$:*

- $\mathsf{Share}^{t,n}(x, r) \xrightarrow{\$} ([x]_1, \ldots, [x]_n)$ *takes input $x \in X$, randomness r and outputs n shares $([x]_1, \ldots, [x]_n)$*
- $\mathsf{Combine}^{t,n}([x]_i, \ldots, [x]_t) \to x'$ *takes a threshold of secret shares $[x]_i$ for $i > t$ as input and combines to form x' the representation of the original message $x' \in X$*

Shamir Secret Sharing

Threshold Public-Key Encryption A Threshold Public-Key Encryption Scheme TPK is a set of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Verify}, \mathsf{Combine})$ over $\mathcal{M}$:

- $\mathsf{TPK.Setup}(1^n, n, t) \xrightarrow{\$} \{\mathsf{pk}, \mathsf{vk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)\}$: input the t of n threshold, output pk the public key, vk the verification key, and sk_i the shared secret key for each party.
- $\mathsf{TPK.Enc}(\mathsf{pk}, m, \rho) \xrightarrow{\$} \beta$: input message m and randomness ρ, output encryption β
- $\mathsf{TPK.Dec}(\beta, \mathsf{sk}_i)) \to m_i$: each party decrypts β with their shared secret key sk_i
- $\mathsf{TPK.Verify}(\mathsf{pk}, \mathsf{vk}, m_i) \to \{0,1\}$: input pk, vk and share of m_i, verify m_i was computed correctly from pk, vk
- $\mathsf{TPK.Combine}(\mathsf{pk}, \mathsf{vk}, m_{i i \in \mathcal{S} \subseteq [n] s.t. |\mathcal{S}| \geq t+1}) \to m$: recovers message m given $t+1$ partial decryptions which verify successfully

5.8 Private VRF

Private VRF/PRF usage is based on the scheme in [TBA+22]. To illustrate the use of a private VRF, a user is assumed to have a verified commitment to the values required, that is $Com_{vrf} = h_1^s h_2^{ctx} w^t$. The user generates a verification key $vk = \tilde{h}^{s+ctx} \tilde{w}^u$, proof $y = e(nullif, \tilde{w})^t$ and their VRF output aka nullifier $h^{\frac{1}{s+ctx}}$.

- $VRF.Gen(s)$: $vk = \tilde{h}^{s+ctx} \tilde{w}^t$
- $VRF.Prove_s(ctx)$: $nullif = h^{\frac{1}{s+ctx}}, y = e(nullif, \tilde{w})^t$
- $VRF.Verify(nullif, vk, y)$ such that $e(nullif, vk) \stackrel{?}{=} e(h, \tilde{h}) \cdot y$

PRF Correctness We argue the correctness of vk and y below, the verifier argues correctness of $nullifier$ via pairings:

$$\begin{aligned} e(nullif, vk) &= e(nullif, \tilde{h}^{s+ctx}\tilde{w}^{t}) \\ &= e(nullif, \tilde{h}^{s+ctx}) \cdot e(nullif, \tilde{w}^{t}) \\ &= e(h^{\frac{1}{s+ctx}}, \tilde{h}^{s+ctx}) \cdot y \\ &= e(h, \tilde{h}) \cdot y \end{aligned}$$

argume correctness of the nullifier in zero knowledge

Correctness of vk and y

$$\mathcal{R}(\mathsf{com}, \mathsf{nullif}, \mathsf{vk}, y, t) \text{ holds:}$$
$$\begin{pmatrix} \mathsf{com} = \mathsf{Com}([s, \mathsf{ctx}]; t) & \wedge \\ \mathsf{vk} = \tilde{h}^{s+ctx}\tilde{w}^{t} & \wedge \\ y = e(nullif, w)^{t} & \end{pmatrix} \tag{1}$$

$$\begin{array}{lcl}
\textbf{Peggy} & & \textbf{Victor} \\
\hline
s, \mathsf{ctx}, t & & \mathsf{com}, \mathsf{vk}, y \\
\hat{s}, \hat{ctx}, \hat{t}, \hat{r} \leftarrow\!\!\$\ \mathbb{Z}_q^4 & & \\
T_1 := h_1^{\hat{s}} h_2^{\hat{ctx}} w^{\hat{t}} & & \\
T_2 := \tilde{h_1}^{s+\hat{ctx}} \tilde{w}^{\hat{r}} & & \\
T_3 := e(nullif, w)^{\hat{t}} & \xrightarrow{\quad T_1, T_2, T_3 \quad} & \\
 & \xleftarrow{\quad c \quad} & c \leftarrow\!\!\$\ \mathbb{Z} \\
z_s = \hat{s} + c \cdot s & & \\
z_{ctx} = \hat{ctx} + c \cdot ctx & & \\
z_t = \hat{t} + c \cdot t & & \\
 & \xrightarrow{\quad z_s, z_{ctx}, z_t \quad} & \\
 & & h_1^{z_s} h_2^{z_{ctx}} w^{z_t} \stackrel{?}{=} \mathsf{com}^c \cdot T_1 \\
 & & \tilde{h_1}^{z_s + z_{ctx}} \tilde{w}^{z_t} \stackrel{?}{=} \mathsf{vk}^c \cdot T_2 \\
 & & e(nullif, w)^{z_t} \stackrel{?}{=} y^c \cdot T_3
\end{array}$$

5.9 Private VRF Usage in our Identity System

A user with registration credential *rcd* and associated commitment rcm wants to generate a new context credential *ccm* while proving to the issuer their *ccm* is sybil.

$$\mathcal{R}(\mathsf{rcm}, \mathsf{ccm}, \mathsf{nullif}, \mathsf{vk}, y, t) \text{ holds:}$$
$$\begin{pmatrix} \mathsf{rcm} = \mathsf{Com}([s]; r_{rcm}) & \wedge \\ \mathsf{ccm} = \mathsf{Com}([ctx]; r_{ccm}) & \wedge \\ \mathsf{vk} = \tilde{h}^{s+ctx} \tilde{w}^{t} & \wedge \\ y = e(nullif, w)^{t} & \end{pmatrix} \tag{2}$$

Peggy		**Victor**
$s, \mathsf{ctx}, t, r_{rcm}, r_{ccm}$		$\mathsf{rcm}, \mathsf{ccm}, \mathsf{vk}, y$
$\hat{s}, \hat{ctx}, r_{\hat{rcm}}, r_{\hat{ccm}}, \hat{t} \leftarrow\$ \mathbb{Z}_q^5$		
$T_1 := h_1^{\hat{s}} w^{r_{\hat{rcm}}}$		
$T_2 := h_1^{\hat{ctx}} w^{r_{\hat{ccm}}}$		
$T_3 := \tilde{h_1}^{\hat{s+ctx}} \tilde{w}^{\hat{t}}$		
$T_4 := e(nullif, w)^{\hat{t}}$	$\xrightarrow{T_1, T_2, T_3, T_4}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \mathbb{Z}$
$z_s = \hat{s} + c \cdot s$		
$z_{ctx} = \hat{ctx} + c \cdot ctx$		
$z_t = \hat{t} + c \cdot t$		
$z_{r_{rcm}} = r_{\hat{rcm}} + c \cdot r_{rcm}$		
$z_{r_{ccm}} = r_{\hat{ccm}} + c \cdot r_{ccm}$		
	$\xrightarrow{z_s, z_{ctx}, z_t, z_{r_{rcm}}, z_{r_{ccm}}}$	
		$h_1^{z_s} w^{z_{r_{rcm}}} \stackrel{?}{=} \mathsf{rcm}^c \cdot T_1$
		$h_2^{z_{ctx}} w^{z_{r_{ccm}}} \stackrel{?}{=} \mathsf{ccm}^c \cdot T_2$
		$\tilde{h_1}^{z_s + z_{ctx}} \tilde{w}^{z_t} \stackrel{?}{=} \mathsf{vk}^c \cdot T_3$
		$e(nullif, w)^{z_t} \stackrel{?}{=} y^c \cdot T_4$

5.10 Random Function

Game $\mathsf{Rand}^{A}_{\{0,1\}^3}\ Fn(x)$	$\mathcal{A}$
if $T[x] = \perp$ then $T[x] \leftarrow\!\$\ R$	$y_1 \leftarrow\!\$\ Fn(00)$
return $T[x]$	$y_2 \leftarrow\!\$\ Fn(11)$
	$return(y_1 \oplus y_2 = 101)$

$$\Pr\left[\mathsf{Rand}^{A}_{\{0,1\}^3} \to true\right] = 2^{-3}$$

5.11 IND-CPA

$$\begin{bmatrix} b \leftarrow\!\$\ \{0,1\} \\ b \leftarrow\!\$\ \{0,1\}\, open \neq 0 \wedge m_0 \neq m_1 : \\ b \leftarrow\!\$\ \{0,1\} \end{bmatrix} \leq \mathsf{negl}(n)(n)$$

IND-CPA$_e nc^a dv$	
$b \leftarrow\!\$\ \{0,1\}$	$b \leftarrow\!\$\ \{0,1\}$
$b \leftarrow\!\$\ \{0,1\}$	$b \leftarrow\!\$\ \{0,1\}$
$b \leftarrow\!\$\ \{0,1\}$	$b \leftarrow\!\$\ \{0,1\}$
$b \leftarrow\!\$\ \{0,1\}$	$b \leftarrow\!\$\ \{0,1\}$

$\mathsf{Adv}^{\mathrm{prf}}_{\mathcal{A},\mathsf{PRF}}(n)$ $\mathsf{Adv}^{\mathrm{prf}}_{\mathcal{A},\mathsf{PRF}}(arg)$

$\mathsf{Adv}^{\mathrm{prf}}_{\mathcal{A},\mathsf{PRF}}(n)$ $\mathsf{Adv}^{\mathrm{rand}}_{\mathcal{A},\mathsf{PRF}}(arg)$ $\Pr\left[\mathsf{Rand}^{A}_{R} \to d\right]$

	$\mathsf{Game}_1(n)$	$\mathsf{Game}_2(n)$
1 :	Step 1	Step 1
2 :	Step 2	Step 2

IND-CPA$_{e}nc^{a}dv$	
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$

$\mathsf{Adv}^{\mathrm{prf}}_{\mathcal{A},\mathsf{PRF}}(n)$ $\mathsf{Adv}^{\mathrm{prf}}_{\mathcal{A},\mathsf{PRF}}(arg)$

$\Pr[X = x]$ $\Pr_{x \leftarrow\$ \{0,1\}^n}[x = 5]$ $\Pr[X = x \mid A = b]$ $\Pr_{x\ sample\ bin^n}[x = 5 \mid A = b]$

First	**Second**
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$

IND-CPA$^{\mathcal{A}}_{\mathsf{Enc}}$
$b \leftarrow\$ \{0,1\}$

IND-CPA$^{\mathcal{A}}_{\mathsf{Enc}}$	
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$

The hiding-advantage of $\mathcal{A}$ is $\mathsf{Adv}^{\mathrm{hide}}_{\mathcal{A},\mathsf{PRF}}(n)$
The hiding-advantage of $\mathcal{A}$ is $\mathsf{Adv}^{\mathrm{hide}}_{Com}(\mathcal{A}) = 2 \cdot \Pr[]$

First	**Second**
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$

IND-CPA$^{\mathcal{A}}_{\mathsf{Enc}}$
$b \leftarrow\$ \{0,1\}$

IND-CPA$^{\mathcal{A}}_{\mathsf{Enc}}$	
$b \leftarrow\$ \{0,1\}$	$b \leftarrow\$ \{0,1\}$

CPA = Adversary picks messages

	$\mathsf{Game}_3(n)$	$\mathsf{Game}_4(n)$
1 :	Step 1	Step 1
2 :	Step 2	Step 2

	$\mathsf{Game}_1(n)$	$\mathsf{Game}_2(n)$
1 :	Step 1	Step 1
2 :	Step 2	Step 2

References

TBA+22. A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai. Utt: Decentralized ecash with accountable privacy. *Cryptology ePrint Archive*, 2022.