Differential Trust: Dynamic Multi-Authority Anonymous Credentials with Epoch-Weighted Updates

Chen Li
Tianjin University
lichen1232022@163.com

Jianting Ning*
Zhejiang Sci-Tech University
jtning88@gmail.com

Xiulong Liu
Tianjin University
xiulong_liu@tju.edu.cn

Yulin Liu
Wuhan University
liuyulin@whu.edu.cn

Abstract

Anonymous credentials (ACs) are fundamental to privacy-preserving authentication, allowing users to prove possession of attributes without revealing their identities. State-of-the-art ACs distribute credential issuance across multiple authorities, typically employing techniques such as Shamir's secret sharing or aggregate signatures. While this approach enhances system robustness and eliminates single point of failure, it treats all authorities equally in the credential issuance phase. This uniform treatment disregards the varying levels of trustworthiness or stake held by different authorities. Such limitation has become particularly problematic in modern decentralized systems like Proof-of-Stake networks, where the inherent trust differentiation among nodes cannot be leveraged in the credential issuance process.

To address this limitation, we propose the notion of **M**ulti-**A**uthority **A**nonymous **C**redentials with **E**poch-Based **W**eights (MA-ACEW), the first Multi-Authority Anonymous Credential (MA-AC) model that considers authorities' weight distribution in credential issuance. Crucially, MA-ACEW enables efficient credential updates when authority weight distributions change across epochs. The core of MA-ACEW is our novel Epoch-Bound Pointcheval-Sanders Signature (EB-PS) primitive, which binds signatures to specific time epochs. This temporal binding enables both weight-based credential issuance within epochs and efficient non-interactive credential updates across epochs. We formalize the EUF-eCMA unforgeability requirement for EB-PS and prove our construction satisfies it under a novel STB-GPS assumption. We then prove that our MA-ACEW construction achieves unforgeability, anonymity, and blindness. Finally, we present benchmarks demonstrating the efficiency of EB-PS and MA-ACEW. Remarkably, presenting a credential aggregated from 128 partial ones takes only 10.68 ms on average. [1]

*Corresponding author. Part of this work was done while the author was at Wuhan University.

[1]We provide additional definitions, detailed proofs, and additional properties in the full version of this paper [53].

1 Introduction

Digital authentication has evolved into a fundamental security primitive bridging digital and physical identities. Such authenticated identities enable secure cross-domain access to real-world services and resources without physical presence requirements. A prominent example is Estonia's *e-Residency* program, launched in 2014, which enables global users from over 170 countries to obtain digital identities for accessing e-government services (e.g., company registration, tax filing), generating over €150 million in direct economic value [34].

However, despite providing convenient service access, this approach raises significant privacy concerns as users' digital identities become vulnerable to unauthorized exposure and potential misuse. At first glance, one might perceive privacy preservation and identity authentication as inherently contradictory objectives: *How can one prove their identity while maintaining privacy?*

Yet, already in the 1980s, Chaum [24], [25] provided an elegant solution to this challenge through his pioneering work on cryptographic techniques for creating privacy-friendly and user-centric authentication solutions. Later, anonymous credentials (ACs) emerged as a cryptographic primitive that enables users to prove specific attributes (such as being over 18) without revealing any additional personal information or creating linkable traces across different authentications. Over the past decades, ACs have attracted significant research attention, resulting in a vast body of research exploring diverse approaches [5, 6, 18–21, 38, 40, 44, 61].

Beyond theoretical research, anonymous credentials have found their way into various practical applications. A notable example is Direct Anonymous Attestation (DAA), which has been integrated into the Trusted Platform Module specification [15] and has seen continued development [52, 68]. More recent applications include PrivacyPass for anonymous web authentication [31], privacy-preserving point collection systems [10], and anonymous tokens [48, 51] for secure authorization.

Despite significant advances in functionality and adoption,

ACs face two critical challenges in today's distributed landscape. First, reliance on a single trusted issuer creates a centralization vulnerability where compromised signing keys could produce unauthorized yet valid credentials. Second, the emergence of blockchain credential platforms such as Hyperledger Indy [37], Veramo [36], and Okapi [64] has created a paradigm shift toward distributed architectures that fundamentally conflicts with centralized issuance models. These challenges motivate the need for distributed anonymous credential systems that maintain traditional security and privacy properties while supporting multiple authorities. Recent research has pursued two approaches: threshold issuance schemes [28, 32, 63] that distribute trust among multiple parties using Shamir's secret sharing [62], and Multi-Authority Anonymous Credentials (MA-ACs) [45, 56] that enable independent issuers to coexist within a single system through aggregate signatures with randomizable tags, allowing users to combine credentials from different sources into compact proofs while preserving privacy.

Even so, these distributed credential schemes face a significant misalignment with the inherent nature of modern distributed networks. In decentralized ecosystems, participants hold different levels of authority and trust, fundamentally shaping network dynamics and consensus mechanisms. However, existing ACs, both threshold-based and multi-authority, treat all partial credentials uniformly, disregarding the inherent trust asymmetry among issuing authorities. This uniform treatment fails to capture the natural heterogeneity of trustworthiness in blockchain governance. For instance, in Proof-of-Stake (PoS) systems [30, 49] or Decentralized Autonomous Organizations (DAOs) [66], a validator with a substantial stake or a governance delegate with high reputational backing inherently warrants greater influence than a peripheral node with minimal investment. Such differential weighting is crucial for security and fairness [60], ensuring that decision-making power accurately reflects the stakeholders' tangible commitment to the ecosystem. Moreover, these power structures are not static but evolve over time, stake distributions fluctuate due to slashing events, delegation updates, or token transfers, requiring dynamic weight adjustments. Addressing this mismatch requires a paradigm shift from static, unweighted credential schemes toward dynamic, weighted mechanisms to foster sustainable decentralized ecosystems. We provide detailed motivation in Section 2. Our analysis of this limitation led us to explore weighted authority models, from which we derive the following research challenge:

How to design an anonymous credential system with weight distribution among different authorities and simultaneously support dynamic weight updates?

While existing works [28,32,56,63] have primarily focused on systems with equal or static weights, our work extends this line of research by considering dynamic weight distribution among authorities. To this end, we introduce an epoch-based mechanism that assigns and refreshes authority weights over time, together with efficient credential updates that ensure only epoch-valid credentials can be verified. Our main contributions are summarized as follows:

- **The EB-PS Primitive**. To enforce temporal constraints on signature validity, we introduce a new primitive called Epoch-Bound Pointcheval-Sanders Signature (EB-PS). In this primitive, the signing process is split into two phases: message signing and epoch-specific binding. Our construction of EB-PS builds upon AtoSa [56], which itself extends the multi-message PS signature scheme [58]. Our key innovation is the introduction of a dynamic management process for one of the secret key components. Specifically, we designate a component, the epoch key z_j, which, unlike the other static key components, must be periodically updated. Upon a transition to a new epoch j', a freshly generated $z_{j'}$ replaces the old z_j. This mandatory key rotation is the core mechanism that binds the signature to a specific time frame. Consequently, a valid EB-PS signature can be viewed as a combination of a long-term signature and a short-term, epoch-specific signature. While preserving the randomization properties of AtoSa, EB-PS achieves remarkable efficiency: a signature aggregated from n messages remains compact at only two group elements.
- **The MA-ACEW Construction**. Based on our EB-PS construction and the Weighted Threshold Signature scheme proposed by Das et al. [29], we present a new AC construction, named Multi-Authority Anonymous Credentials with Epoch-Based Weights (MA-ACEW). To the best of our knowledge, MA-ACEW is the first anonymous credential system that enables weighted trust evaluation across multiple authorities (i.e., multiple issuers). MA-ACEW addresses practical scenarios where each issuer is assigned different weights according to the issuer's trust level or computational capability. Additionally, the redistribution or adjustment of issuer weights is allowed during epoch transitions. For credential updates during epoch transitions, we eliminate the need for users to re-interact with each issuer. Instead, users perform a one-time interactive obtaining process with an issuer, and subsequent operations involve the issuer computing epoch-specific partial credentials for users holding long-term credentials. When a user wants to present their credential to a verifier during epoch j, they only need to compute the corresponding proofs and combine the partial credentials into one. Finally, we extend the core framework to natively support three crucial features: (i) multi-attribute credentials, (ii) issuer hiding, and (iii) selective disclosure of attributes under arbitrary predicates. Collectively, these properties make MA-ACEW particularly suitable for practical scenarios, especially in distributed networks.
- **New Assumption and Formal Security Proofs**. We introduce and formalize a new hardness assumption, the Separable Time-Bound Generalized PS (STB-GPS) assumption. This assumption extends the well-established Generalized

PS (GPS) assumption [50, 58] to more accurately capture the security requirements of epoch-based signature schemes where signature components are separable and time-bound, and we analyze its hardness in the Generic Group Model (GGM). We then establish a formal security model for our epoch-based primitive, defining Existential Unforgeability under chosen-message and Epoch-corruption Attack (EUF-eCMA). This model is formulated in the chosen-key setting [13, 54, 56] and is specifically designed to handle the dynamics of epoch transitions and, crucially, allows the adversary to perform epoch-secret corruptions. We then rigorously prove that our EB-PS construction achieves EUF-eCMA security under the STB-GPS assumption. Furthermore, for our MA-ACEW construction, we provide formal security definitions for its three key properties: unforgeability, anonymity, and blindness. Building upon the security models from [38], [45], and [56], our models introduce additional oracles to capture epoch transitions and partial credential issuance, which are crucial for our system. We then provide rigorous proofs demonstrating that MA-ACEW satisfies all these specified security features.

- **Implementation**. We implement both the EB-PS and MA-ACEW constructions in Golang. Additionally, we develop a smart contract for credential issuance and verification operations in MA-ACEW. For credential presentation, users need just 10.68 ms to show credentials aggregated from 128 partials. On-chain verification requires 1077K gas on Ethereum with pre-computed $\mathbb{G}_2$ exponentiation.

2 Problem Statement

2.1 Problem Description

In modern decentralized ecosystems, PoS has established itself as the fundamental mechanism for securing trust and consensus. Unlike traditional identity-based systems, authority in a PoS setting is derived strictly from economic backing, where entities exercise power for governance or resource allocation proportional to their held stake. A valid authorization is defined by accumulating a super-majority of the total stake, rather than a simple majority of entities. However, when applying decentralized privacy-preserving credentials to this setting, a structural misalignment arises. Existing decentralized anonymous credentials are inherently stake-agnostic, they treat every issuer's signature as identical, disregarding the fact that authority in PoS is strictly stake-dependent. Consequently, these schemes limit the applicability of privacy-preserving mechanisms in scenarios that require fine-grained, stake-based governance. To bridge this gap, it is necessary to construct a credential system that is compatible with the PoS trust model while strictly preserving user privacy. However, realizing such a system presents two core challenges:

A.1: *The scheme must support a trust model where issuers possess differential weights corresponding to their specific stake or institutional credibility.*

A.2: *The scheme must support dynamic weight adjustments to reflect that issuer trust levels change over time, allowing weights to increase or decrease in response to changes in stake or confidence.*

Limitation of Existing Works. Current distributed anonymous credential systems adopt two main approaches. The first approach, threshold issuance for ACs [32, 63], typically employs a (t, n) Shamir's secret sharing scheme [62], where each of the n issuers holds a share of the signing key, and any subset of t issuers can collaboratively generate a credential. Alternatively, MA-ACs [45, 56] are based on aggregate signatures with randomizable tags, allowing users to aggregate showings of credentials from different issuers (with respect to the same tag) into one compact showing, and can be viewed as a multi-signature scheme. While these approaches effectively distribute the issuing responsibility, they all adopt a uniform-weight paradigm, assigning equal importance to all credential issuers.

Naive Solutions. A straightforward approach to adapt existing ACs to support arbitrary weights is through virtualization. In this model, an issuer with weight w simply holds w distinct signing keys and emulates w separate virtual issuers. However, this method suffers from several critical limitations: (i) The signing cost, partial signature size, and user's computational load all scale linearly with the total weight W, imposing severe scalability constraints. (ii) Any modification to the weight vector $\boldsymbol{w}$ necessitates costly key regeneration and credential re-issuance across the entire system. (iii) Requiring issuers to maintain numerous signing keys significantly increases system complexity and the risk of key compromise.

The severe scalability issues of virtualization, particularly the linear growth in cost and data size, naturally lead to considering more specialized solutions like Weighted Secret Sharing (WSS) schemes. While the concept dates back to Shamir's work [62], a notable recent development is the Weighted Ramp Secret Sharing (WRSS) scheme by Garg et al. [39]. Their construction, which is based on the Chinese Remainder Theorem (CRT), appears highly promising as it directly addresses the primary scaling limitation of virtualization: the share size for a party with weight w is only $O(w)$ bits. This efficiency is achieved in the ramp setting [9], which allows for a gap between the privacy and reconstruction thresholds, seemingly providing a direct path to our desired functionality.

Despite its elegance in solving the scaling problem, this solution is ultimately not a robust one. The core efficiency of WRSS is predicated on a fundamental constraint: its reliance on a ramp setting. This introduces a predefined gap between the reconstruction threshold T, the minimum combined weight of participants needed to recover the secret, and the privacy threshold t, the maximum combined weight that is guaranteed to learn no information about it. This gap (i.e.,

$T - t = \Omega(\lambda)$), while enabling efficiency, also introduces its own severe limitations for practical systems: i) Its reconstruction is set-dependent, meaning the algorithm to recover the secret changes based on the exact set of participants. This requires a preliminary coordination step to identify all active members, creating a bottleneck that is impractical for dynamic or decentralized networks. ii) It introduces security vulnerabilities for small weights, as an $O(w)$-bit share can be discovered via a brute-force attack if a party's weight w is small. iii) It imposes a rigid and static weight structure, since a party's weight is intrinsically tied to its public parameter, and any change requires a costly, system-wide reset.

Summary of Challenges. In summary, naive approaches are unsuitable for dynamic ACs: virtualization incurs prohibitive linear costs, while CRT-based WRSS rigidly binds weights to public parameters. Neither supports weight changes without costly resets. A practical solution must enable efficient updates and decouple weights from credentials—our work is the first to achieve both.

2.2 Solution Overview

Here we present a high-level overview of our solution, which is efficient and conceptually simple. To construct our anonymous credential system, we partition the system into epochs. Each epoch typically spans a fixed duration (e.g., one day or one week), during which issuers' weights remain constant. Our approach is structured into two phases: embedding time epochs into the signature scheme, followed by integrating weight settings based on this modified signature scheme.

Basic Signature Scheme. Our anonymous credential system leverages the Pointcheval-Sanders (PS) signature scheme. In its general form, a signer can sign a vector of messages $(m_1, \ldots, m_n)$ using a secret key $\mathsf{sk} = (x, y_1, \ldots, y_n)$. A signature is a pair of group elements $\sigma = (h, h^{x+\sum_{i=1}^{n} y_i m_i})$ for a random $h \in \mathbb{G}$. This structure supports efficient randomization for anonymity: a user can re-randomize a signature using a random $r \in \mathbb{F}_p$ to obtain $\sigma' = (h^r, (h^{x+\sum_{i=1}^{n} y_i m_i})^r)$, which is a valid signature that cannot be linked to the original.

Our innovation is to enable dynamic credential updates by partitioning the secret key vector. One component, z_j, serves as the epoch-specific key for epoch j, while the remaining $(y_1, \ldots, y_k)$ sign user attributes $(m_1, \ldots, m_k)$. The update protocol works as follows:

1. **Key Updates:** At each epoch j, the issuer replaces z_{j-1} with z_j, keeping $(x, y_1, \ldots, y_k)$ unchanged.

2. **Signature Updates:** With a small public update token, users transform prior signatures into ones valid for the new epoch without re-issuance.

To build intuition for our construction, we begin by describing a basic version. In this simplified exposition, the signature base is the fundamental public parameter h, devoid of any user-specific tag, and the credential contains only a single attribute m. The signature form thus simplifies to $\sigma = (h, h^{x+y \cdot m + z_j \cdot F(\mathsf{ctx}, j)})$. The term $z_j \cdot F(\mathsf{ctx}, j)$ functions as a pseudo random function (PRF) output to provide domain separation for different credential contexts (identified by ctx).

While the element h is a randomly chosen parameter in the original PS signature scheme, it is often adapted in ACs to serve as a user-specific base. Our design leverages this concept by having the issuer store this base h after a user's initial enrollment. Building on this, the issuer can unilaterally issue credential updates for each epoch j by computing and distributing the temporal component $h^{F(\mathsf{ctx}, j) \cdot z_j}$. The exponent is carefully constructed: the epoch-specific secret z_j ensures the value is fresh for each period, while the function F binds the update to the specific context ctx and epoch j. Consequently, as z_j is regenerated for each epoch, the credential update for users is a simple, non-interactive process that only requires fetching this new component.

Incorporating weight setting. Now we first consider the weight distribution across all issuers. In each epoch j, we denote this distribution by a vector $\boldsymbol{w}_j$. Weight vector adjustments occur only during epoch transitions, based on various factors beyond the scope of this paper.

We note that each issuer is assigned a specific weight under epoch j. Consider a set of n issuers, we use a bit vector $\boldsymbol{b}$ to represent the participating issuers in the multi-issuer credential cred for a user, where $b[i] = 1$ denotes participation and $b[i] = 0$ represents non-participation. Consequently, the total weight of the credential cred can be expressed as $\mathrm{W} = \langle \boldsymbol{w}_j, \boldsymbol{b} \rangle$, which is the inner product of vectors $\boldsymbol{w}_j$ and $\boldsymbol{b}$. Recall that each issuer's secret key is composed of (x_i, y_i, z_i), where y_i and z_i are used for signing a message and incorporating epoch information, respectively. The public key component $\mathrm{X}_i = v^{x_i}$ corresponds to private key element x_i and correlates with weight $w_{i,j}$. Thus, $\mathrm{X} = \langle \boldsymbol{pk}_x, \boldsymbol{b} \rangle$ represents the aggregated keys of participating issuers, where $\boldsymbol{pk}_x = (\mathrm{X}_1, \ldots, \mathrm{X}_n)$.

To prove the validity of a weight-based credential, we simultaneously demonstrate three crucial properties: i) the user's aggregated credential is validly obtained from the participating issuers represented by the vector $\boldsymbol{b}$; ii) the user's provided combined weight sum W exceeds the threshold defined by the verifier; and iii) the corresponding aggregated verification key X is correctly computed using the same vector $\boldsymbol{b}$. By verifying these properties, we confirm both the credential's validity and its compliance with the weight requirements, where issuers are weighted rather than treated equally.

For the proofs of inner products $\langle \boldsymbol{pk}_x, \boldsymbol{b} \rangle$ and $\langle \boldsymbol{w}_j, \boldsymbol{b} \rangle$, Das et al. [29] provide an elegant solution that substantially reduces verification complexity. We will demonstrate in Section 5 how to integrate their approach with several minor but important modifications tailored to our construction. During epoch transitions, our solution requires recomputing only the critical components $(h^{F(\mathsf{ctx}, j') \cdot z_{j'}})$ and the proof $\langle \boldsymbol{w}_{j'}, \boldsymbol{b} \rangle$, which significantly reduces the computational overhead associated

with authority set updates.

3 Preliminaries

3.1 Assumption

Definition 1 (Separable Time-Bound Generalized PS (STB-GPS) Assumption)**.** *Given an asymmetric pairing setting $\mathcal{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, u, v, e)$, a challenger chooses random master secrets $x, y \in \mathbb{F}_p$ and a set of independent random epoch secrets $\{z_j\}_{j=1}^{T} \subset \mathbb{F}_p$. The challenger initializes empty lists $\mathcal{Q}_h, \mathcal{Q}_{sign}, \mathcal{Q}_{corrupt}$. An adversary $\mathcal{A}$ is given the public parameters (u, v, u^y, v^x, v^y) and access to the following oracles:*

- ***Oracle*** $\mathcal{O}_h(\cdot)$***:*** *Outputs a uniformly distributed element $h \in \mathbb{G}_1$ and adds h to a list $\mathcal{Q}_h$.*
- ***Oracle*** $\mathcal{O}_{sign}(j, m, h, \mathsf{ctx})$***:*** *If $h \notin \mathcal{Q}_h$, or $j \in \mathcal{Q}_{corrupt}$, or the identity (m, h, ctx) has already been signed for any epoch (i.e., $(m, h, \mathsf{ctx}, \star) \in \mathcal{Q}_{sign}$), it returns $\perp$. Otherwise, it computes:*

$$s_{\mathsf{lt}} = h^{x + m \cdot y}, s_{\mathsf{ep},j} = h^{F(\mathsf{ctx}, j) \cdot z_j}$$

  It adds the tuple $(m, h, \mathsf{ctx}, j, s_{\mathsf{lt}}, s_{\mathsf{ep},j})$ to $\mathcal{Q}_{sign}$ and returns $(s_{\mathsf{lt}}, s_{\mathsf{ep},j})$.
- ***Oracle*** $\mathcal{O}_{update}(j, m, h, \mathsf{ctx})$***:*** *Upon input (j, m, h, ctx), if the tuple for epoch j, $(m, h, \mathsf{ctx}, j, \star)$, is not in $\mathcal{Q}_{sign}$, or if a tuple for epoch $j+1$ already exists, or if the target epoch $j+1$ is corrupted (i.e., $j+1 \in \mathcal{Q}_{corrupt}$), the oracle returns $\perp$. Otherwise, it computes $s_{\mathsf{ep},j+1} \leftarrow h^{F(\mathsf{ctx}, j+1) \cdot z_{j+1}}$. It finds the corresponding long-term part s_{lt} from the record for epoch j, adds the new tuple $(m, h, \mathsf{ctx}, j+1, s_{\mathsf{lt}}, s_{\mathsf{ep},j+1})$ to $\mathcal{Q}_{sign}$, and returns $s_{\mathsf{ep},j+1}$.*
- ***Oracle*** $\mathcal{O}_{corrupt}(j)$***:*** *Returns the secret key z_j for epoch j and adds j to the corruption list $\mathcal{Q}_{corrupt}$.*

The STB-GPS assumption holds if for any PPT adversary $\mathcal{A}$, the probability of outputting a valid forgery tuple $(j^*, m^*, h^*, \mathsf{ctx}^*, s^*_{\mathsf{lt}}, s^*_{\mathsf{ep},j^*})$ is negligible. A tuple is a valid forgery if it satisfies all of the following conditions:

$$\begin{cases} e(s^*_{\mathsf{lt}}, v) = e(h^*, v^{x + m^* y}), \quad e(s^*_{\mathsf{ep},j^*}, v) = e(h^*, v^{F(\mathsf{ctx}^*, j^*) z_{j^*}}) \\ h^* \in \mathcal{Q}_h, \;\; j^* \notin \mathcal{Q}_{corrupt} \\ (m^*, h^*, \mathsf{ctx}^*, j^*, s^*_{\mathsf{lt}}, s^*_{\mathsf{ep},j^*}) \notin \mathcal{Q}_{sign} \end{cases}$$

Theorem 3.1. The STB-GPS assumption holds in the generic group model. After an adversary makes a total of q queries to the assumption's oracles $(\mathcal{O}_h, \mathcal{O}_{\text{sign}}, \mathcal{O}_{\text{update}}, \mathcal{O}_{\text{corrupt}})$ and q_G queries to the group operation oracles, its probability of producing a valid forgery is no more than $(2 + q + q_G)^2 / p$, where p is the prime order of the groups.

The proof is provided in the full paper [53, App. D.1].

3.2 Weighted Threshold Signature

Our work builds on the weighted threshold signature (WTS) scheme by Das et al. [29], which is based on the inner product arguments of Campanelli et al. [23]. For our purposes, we adapt the original construction with a key structural modification. Specifically, we decompose the monolithic Combine algorithm into three more granular components: CombPk for public key aggregation, CombWt for weight combination, and MergePf for aggregating proofs. The complete formal definition of our adapted scheme is provided in the full paper [53, App. B.1].

4 Epoch-Bound Pointcheval-Sanders Signature

4.1 Syntax and Security Definitions

We now introduce the syntax of EB-PS, which extends the framework proposed by Mir et al. [56]. Formally, an EB-PS consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, \mathrm{T}) \to \mathsf{pp}$: On input the security parameter λ and the total number of time periods T, this algorithm outputs the public parameters pp.

$\mathsf{KGen}(\mathsf{pp}, n) \to \{\mathsf{lsk}_i, \mathsf{lvk}_i\}_{i \in [n]}$: On input the public parameters pp and the total number of signers n, the algorithm outputs a set of long-term key pairs $\{\mathsf{lsk}_i, \mathsf{lvk}_i\}_{i \in [n]}$.

$\mathsf{TKGen}(\mathsf{pp}, n, j) \to \{\mathsf{tsk}_{i,j}, \mathsf{tvk}_{i,j}\}_{i \in [n]}$: On input the public parameters pp, the total number of signers n, and an epoch index $j \in [\mathrm{T}]$, it outputs a set of n epoch-specific key pairs where $\mathsf{tsk}_{i,j}$ denotes the signing key and $\mathsf{tvk}_{i,j}$ denotes the verification key for signer i under epoch j.

$\mathsf{GenAuxTag}(\mathrm{S}) \to (\mathsf{aux}, \mathsf{tg})$: On input a message-key set S, the algorithm outputs auxiliary information aux associated with set S and a tag tg.

$\mathsf{SignLt}(\mathsf{lsk}_i, m_i, \mathsf{aux}, \mathsf{tg}) \to \sigma_{\mathsf{lt},i}$: On input the long-term secret key lsk_i, message m_i, auxiliary data aux, and tag tg, the algorithm outputs a long-term signature $\sigma_{\mathsf{lt},i}$.

$\mathsf{SignEp}(j, \mathsf{tg}, \mathsf{tsk}_{i,j}, F(\mathsf{ctx}, j)) \to \sigma_{\mathsf{ep},i,j}$: Under epoch j, given a tag tg, an epoch-specific secret key $\mathsf{tsk}_{i,j}$, and a time-dependent function evaluation $F(\mathsf{ctx}, j)$ where ctx remains constant across epochs and serves solely for epoch binding, the algorithm outputs an epoch-specified signature $\sigma_{\mathsf{ep},i,j}$.

$\mathsf{CombSig}(j, \mathsf{tg}, \sigma_{\mathsf{lt},i}, \sigma_{\mathsf{ep},i,j}) \to \sigma_{i,j}$: Under epoch j, and the same tg, given a long-term signature $\sigma_{\mathsf{lt},i}$ and an epoch-specified signature $\sigma_{\mathsf{ep},i,j}$ for signer i, the algorithm outputs a combined signature $\sigma_{i,j}$ under epoch j.

$\mathsf{Verify}(j, \mathsf{tg}, \mathsf{lvk}_i, \mathsf{tvk}_{i,j}, m_i, F(\mathsf{ctx}, j), \sigma_{i,j}) \to \{0, 1\}$: Under epoch j, given a tag tg, long-term verification key lvk_i, an epoch-specified verification key $\mathsf{tvk}_{i,j}$, message m_i, the common time-dependent function evaluation $F(\mathsf{ctx}, j)$, and a combined signature $\sigma_{i,j}$ for signer i, the algorithm outputs 1 if the

signature is valid and 0 otherwise.

$\mathsf{AggSigLt}(\mathsf{tg}, \{(\mathsf{lvk}_i, m_i, \sigma_{\mathsf{lt},i})\}_{i=1}^{\ell}) \to (\sigma_{\mathsf{agg,lt}}, \mathbb{M}, \mathsf{avk}_{\mathsf{lt}})$: Under the same tag tg, given a set of ℓ long-term signatures $\sigma_{\mathsf{lt},i}$ for the messages $\{m_i\}_{i\in[\ell]}$ under the verification keys $\{\mathsf{lvk}_i\}_{i\in[\ell]}$, the algorithm outputs an aggregate signature $\sigma_{\mathsf{agg,lt}}$, a message set $\mathbb{M}$, and an aggregated verification key $\mathsf{avk}_{\mathsf{lt}}$.

$\mathsf{AggSigEp}(j, \mathsf{tg}, \{\mathsf{tvk}_{i,j}, \sigma_{\mathsf{ep},i,j}\}_{i=1}^{\ell}, F(\mathsf{ctx}, j)) \to (\sigma_{\mathsf{agg,ep},j}, \mathsf{avk}_{\mathsf{ep},j})$: Under epoch j and the same tg, given a set of ℓ epoch-specified signatures $\sigma_{\mathsf{ep},i,j}$ under the verification keys $\{\mathsf{tvk}_{i,j}\}_{i\in[\ell]}$ with respect to the common time-dependent function evaluation $F(\mathsf{ctx}, j)$, the algorithm outputs an aggregate epoch signature $\sigma_{\mathsf{agg,ep},j}$ and an aggregated epoch verification key $\mathsf{avk}_{\mathsf{ep},j}$.

$\mathsf{CombAggSig}(j, \mathsf{tg}, \sigma_{\mathsf{agg,lt}}, \sigma_{\mathsf{agg,ep},j}) \to \sigma_{\mathsf{agg},j}$: Under epoch j and the same tg, given an aggregated long-term signature $\sigma_{\mathsf{agg,lt}}$ and an aggregated epoch signature $\sigma_{\mathsf{agg,ep},j}$, the algorithm outputs a combined aggregate signature $\sigma_{\mathsf{agg},j}$.

$\mathsf{AggVerify}(j, \mathsf{tg}, \mathsf{avk}_j, \mathbb{M}, \sigma_{\mathsf{agg},j}, F(\mathsf{ctx}, j)) \to \{0,1\}$: Under epoch j, given a tag tg, an aggregated verification key $\mathsf{avk}_j = (\mathsf{avk}_{\mathsf{lt}}, \mathsf{avk}_{\mathsf{ep},j})$, a message set $\mathbb{M}$, a combined aggregate signature $\sigma_{\mathsf{agg},j}$, and the common time-dependent function evaluation $F(\mathsf{ctx}, j)$, the algorithm outputs 1 if the aggregate signature is valid and 0 otherwise.

$\mathsf{RndSigTag}(\mathsf{avk}_j, \mathsf{tg}, \sigma_{\mathsf{agg},j}, r) \to (\sigma'_{\mathsf{agg},j}, \mathsf{tg}')$: Under aggregated verification key avk_j, given a tag tg, an aggregate signature $\sigma_{\mathsf{agg},j}$, and a random value r, the algorithm outputs a randomized aggregate signature $\sigma'_{\mathsf{agg},j}$ and a randomized tag tg'.

Correctness. EB-PS ensures both basic and aggregation correctness, as formalized in the full paper [53, App. C.1].

Unforgeability. To formalize the security of our epoch-based scheme, we work within the chosen-key model of security, following the line of work in [13, 54, 56]. Our security notion, Existential Unforgeability under chosen-message and Epoch-corruption Attack (EUF-eCMA), by building an interactive game where the adversary is given a single challenge public key vk' and access to a corresponding signing oracle.

Definition 2 (Existential Unforgeability under chosen-message and Epoch-corruption Attack (EUF-eCMA))**.** *An epoch-bound digital signature scheme* $\Sigma = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{TKGen}, \mathsf{SignLt}, \mathsf{SignEp}, \mathsf{AggVerify})$ *is EUF-eCMA secure if for any PPT adversary* $\mathcal{A}$*, the probability of winning the following game is negligible.*

Setup. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathrm{T})$ to generate the public parameters. Then, $\mathcal{C}$ generates a long-term key pair $(\mathsf{lvk}^*, \mathsf{lsk}^*) \leftarrow \mathsf{KGen}(\mathsf{pp})$, where the secret key lsk^* is never revealed. For each epoch $j \in [1, \mathrm{T}]$, $\mathcal{C}$ computes the epoch-specified key pair $(\mathsf{tvk}_j^*, \mathsf{tsk}_j^*) \leftarrow \mathsf{TKGen}(\mathsf{pp}, j)$. $\mathcal{C}$ initializes an empty query log $\mathcal{Q} \leftarrow \emptyset$ and an empty set of corrupted epochs $\mathcal{Q}_{CE} \leftarrow \emptyset$. Finally, $\mathcal{C}$ provides the adversary $\mathcal{A}$ with the public parameters pp, the long-term public key lvk^*, and all epoch public keys $\{\mathsf{tvk}_j^*\}_{j\in[1,\mathrm{T}]}$.

Queries. The adversary $\mathcal{A}$ can adaptively issue the following queries:

- **Initial Signing Query:** On input an epoch $j \in [1, \mathrm{T}]$, a tag tg, a message m, auxiliary info aux, and a context ctx:
  - If epoch $j \in \mathcal{Q}_{CE}$, return $\perp$.
  - Else compute $\sigma_{\mathsf{lt}} \leftarrow \mathsf{SignLt}(\mathsf{lsk}^*, m, \mathsf{aux}, \mathsf{tg})$ and $\sigma_{\mathsf{ep},j} \leftarrow \mathsf{SignEp}(j, \mathsf{tg}, \mathsf{tsk}_j^*, F(\mathsf{ctx}, j))$.
  - Return $(\sigma_{\mathsf{lt}}, \sigma_{\mathsf{ep},j})$ to $\mathcal{A}$ and update $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(j, \mathsf{tg}, m)\}$.
- **Epoch Update Query**: For a transition from epoch j to $j+1$ ($j < \mathrm{T}$), on input a tag tg and context ctx:
  - If $j+1 \in \mathcal{Q}_{CE}$, return $\perp$.
  - Otherwise, compute $\sigma_{\mathsf{ep},j+1} \leftarrow \mathsf{SignEp}(j+1, \mathsf{tg}, \mathsf{tsk}_{j+1}^*, F(\mathsf{ctx}, j+1))$ and return it to $\mathcal{A}$.
- **Epoch Corruption Query**: On input an epoch index $j \in [1, \mathrm{T}]$, add j to $\mathcal{Q}_{CE}$ and return tsk_j^*.

Winning Condition. The adversary $\mathcal{A}$ outputs a forgery tuple $(j^*, \mathsf{tg}^*, \mathbb{M}^*, \sigma^*_{\mathsf{agg},j^*}, \mathsf{avk}_{j^*})$, where $\mathbb{M}^* = \{m_k^*\}_{k=1}^{\ell}$. $\mathcal{A}$ wins if all of the following conditions hold:

$$\begin{cases} \mathsf{AggVerify}(j^*, \mathsf{tg}^*, \mathsf{avk}_{j^*}, \mathbb{M}^*, \sigma^*_{\mathsf{agg},j^*}, F(\mathsf{ctx}, j^*)) = 1 \\ (\mathsf{lvk}^*, \mathsf{tvk}^*_{j^*}) \in \mathsf{avk}_{j^*}, \quad j^* \notin \mathcal{Q}_{CE} \\ \exists m_k^* \in \mathbb{M}^* \text{ s.t. } (j^*, m_k^*, \mathsf{tg}^*) \notin \mathcal{Q} \end{cases}$$

4.2 The EB-PS Construction

In this subsection, we present our EB-PS construction. Inspired by [28, 56], we derive the auxiliary data aux by hashing relevant information to ensure a uniform h component. Following [56], we adopt a unique tag tg for partial aggregation, requiring all partial signatures to share the same tag. The EB-PS scheme proceeds as follows:

$\mathsf{Setup}(1^\lambda, \mathrm{T}) \to \mathsf{pp}$: On input a security parameter λ and the number of epochs T, this algorithm generates bilinear groups $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, u, v, e) \leftarrow \mathsf{BGGen}(1^\lambda)$, where p is a prime order, u is a generator of $\mathbb{G}_1$, and v is a generator of $\mathbb{G}_2$. Then, it selects a hash function $\mathrm{H} : \{0,1\}^* \to \mathbb{G}_1$ and outputs the public parameters $\mathsf{pp} = \{\mathsf{BG}, \mathrm{T}, \mathrm{H}\}$.

$\mathsf{KGen}(\mathsf{pp}, n) \to \{\mathsf{lsk}_i, \mathsf{lvk}_i\}_{i\in[n]}$: For the i-th signer, this algorithm randomly samples $(x_i, y_i) \in \mathbb{F}_p^2$, sets the long-term secret key as $\mathsf{lsk}_i = (x_i, y_i)$ and computes the corresponding verification key as $\mathsf{lvk}_i = (\mathrm{X}_i, \mathrm{Y}_i) = (v^{x_i}, v^{y_i})$.

$\mathsf{TKGen}(\mathsf{pp}, n, j) \to \{\mathsf{tsk}_{i,j}, \mathsf{tvk}_{i,j}\}_{i\in[n]}$: For epoch $j \in [\mathrm{T}]$ and the i-th signer, this algorithm randomly samples $z_{i,j} \in \mathbb{F}_p$, sets the epoch-specific secret key as $\mathsf{tsk}_{i,j} = z_{i,j}$ and computes the corresponding verification key as $\mathsf{tvk}_{i,j} = \mathrm{Z}_{i,j} = v^{z_{i,j}}$.

$\mathsf{GenAuxTag}(\mathrm{S}) \to (\mathsf{aux}, \mathsf{tg})$: Given a message-key set $\mathrm{S} = \{(m_i, \mathsf{lvk}_i)_{i\in[\ell]}\}$, the algorithm randomly samples $(\gamma, \delta) \in \mathbb{F}_p^2$, sets $\mathsf{aux} = u^\gamma \parallel u^\delta \parallel \{(m_i, \mathsf{lvk}_i)\}_{i\in[\ell]}$, computes $h = \mathrm{H}(\mathsf{aux})$,

and outputs auxiliary data aux and tag $\mathsf{tg} = (\Gamma = h^{\gamma}, \Delta = h^{\delta})$, where all verification keys lvk_i must be distinct.

$\mathsf{SignLt}(\mathsf{lsk}_i, m_i, \mathsf{aux}, \mathsf{tg}) \to \sigma_{\mathsf{lt},i}$: Given a long-term secret key $\mathsf{lsk}_i = (x_i, y_i)$ for the i-th signer, message m_i, auxiliary data aux, and tag tg, the algorithm checks that aux has the form $(u^{\gamma} \parallel u^{\delta} \parallel \{(m_i, \mathsf{lvk}_i)\}_{i\in[\ell]})$, and verifies that $(m_i, \mathsf{lvk}_i) \in \mathsf{aux}$; if not, it outputs $\perp$. Otherwise, the algorithm computes $\sigma_{\mathsf{lt},i} = (h', s_{\mathsf{lt},i})$ where:

$$h' = h^{\gamma}, \quad s_{\mathsf{lt},i} = (h^{\gamma})^{x_i + m_i \cdot y_i}$$

$\mathsf{SignEp}(j, \mathsf{tg}, \mathsf{tsk}_{i,j}, F(\mathsf{ctx}, j)) \to \sigma_{\mathsf{ep},i,j}$: Under the epoch j, given a tag tg, an epoch-specific secret key $\mathsf{tsk}_{i,j} = z_{i,j}$ for the i-th signer, and a time-dependent function evaluation $F(\mathsf{ctx}, j)$, the algorithm outputs:

$$\sigma_{\mathsf{ep},i,j} = s_{\mathsf{ep},i,j} = (h^{\delta})^{F(\mathsf{ctx},j)\cdot z_{i,j}}$$

$\mathsf{CombSig}(j, \mathsf{tg}, \sigma_{\mathsf{lt},i}, \sigma_{\mathsf{ep},i,j}) \to \sigma_{i,j}$: Under epoch j and the same tag tg, given a long-term signature $\sigma_{\mathsf{lt},i}$, and an epoch-specific signature $\sigma_{\mathsf{ep},i,j}$, the algorithm outputs a combined signature $\sigma_{i,j}$ computed as:

$$\sigma_{i,j} = (h', s_i = s_{\mathsf{lt},i} \cdot s_{\mathsf{ep},i,j})$$

$\mathsf{Verify}(j, \mathsf{tg}, \mathsf{lvk}_i, \mathsf{tvk}_{i,j}, m_i, F(\mathsf{ctx}, j), \sigma_{i,j}) \to \{0, 1\}$: Under epoch j, given a tg, a long-term verification key $\mathsf{lvk}_i = (\mathrm{X}_i, \mathrm{Y}_i)$ and an epoch-specific verification key $\mathsf{tvk}_{i,j} = \mathrm{Z}_{i,j}$ for the i-th signer, messages m_i and $F(\mathsf{ctx}, j)$, and a signature $\sigma_{i,j}$, this algorithm parses $\sigma_{i,j}$ as (h', s_i) and outputs 1 if the equation holds:

$$e(h', \mathrm{X}_i \cdot \mathrm{Y}_i^{m_i}) e((h^{\delta})^{F(\mathsf{ctx},j)}, \mathrm{Z}_{i,j}) = e(s_i, v) \wedge h' \neq 1_{\mathbb{G}}$$

$\mathsf{AggSigLt}(\mathsf{tg}, \{(\mathsf{lvk}_i, m_i, \sigma_{\mathsf{lt},i})\}_{i=1}^{\ell}) \to (\sigma_{\mathsf{agg,lt}}, \mathbb{M}, \mathsf{avk}_{\mathsf{lt}})$: Under the same tg, given a set of ℓ long-term signatures $\sigma_{\mathsf{lt},i}$ for the messages $\{m_{1,i}\}_{i\in[\ell]}$ under the verification keys $\{\mathsf{lvk}_i\}_{i\in[\ell]}$, this algorithm outputs the set of messages $\mathbb{M} = \{m_i\}_{i=1}^{\ell}$, the aggregated verification key $\mathsf{avk}_{\mathsf{lt}} = \{\mathsf{lvk}_i\}_{i\in[\ell]}$, and an aggregate long-term signature:

$$\sigma_{\mathsf{agg,lt}} = (h', \prod_{i=1}^{\ell} s_{\mathsf{lt},i})$$

$\mathsf{AggSigEp}(j, \mathsf{tg}, \{\mathsf{tvk}_{i,j}, \sigma_{\mathsf{ep},i,j}\}_{i=1}^{\ell}, F(\mathsf{ctx}, j)) \to (\sigma_{\mathsf{agg,ep},j}, \mathsf{avk}_{\mathsf{ep},j})$: Under epoch j and the same tg, given a set of ℓ epoch-specified signatures $\sigma_{\mathsf{ep},i,j}$ under the verification keys $\{\mathsf{tvk}_{i,j}\}_{i\in[\ell]}$ with respect to the common time-dependent function evaluation $F(\mathsf{ctx}, j)$, this algorithm outputs the aggregated epoch verification key $\mathsf{avk}_{\mathsf{ep},j} = \{\mathsf{tvk}_{i,j}\}_{i\in[\ell]}$ and an aggregate epoch signature:

$$\sigma_{\mathsf{agg,ep},j} = \prod_{i=1}^{\ell} s_{\mathsf{ep},i,j}$$

$\mathsf{CombAggSig}(j, \mathsf{tg}, \sigma_{\mathsf{agg,lt}}, \sigma_{\mathsf{agg,ep},j}) \to \sigma_{\mathsf{agg},j}$: Under epoch j and the same tg, given an aggregated long-term signature $\sigma_{\mathsf{agg,lt}}$ and an aggregated epoch signature $\sigma_{\mathsf{agg,ep},j}$, this algorithm combines them to output an aggregate signature $\sigma_{\mathsf{agg},j}$, where:

$$\sigma_{\mathsf{agg},j} = \left(h', s = \sigma_{\mathsf{agg,lt}} \cdot \sigma_{\mathsf{agg,ep},j}\right)$$

$\mathsf{AggVerify}(j, \mathsf{tg}, \mathsf{avk}_j, \mathbb{M}, \sigma_{\mathsf{agg},j}, F(\mathsf{ctx}, j)) \to \{0, 1\}$: Under epoch j, given an aggregate verification key $\mathsf{avk} = \{(\mathsf{lvk}_i, \mathsf{tvk}_{i,j})\}_{i=1}^{\ell}$ where $\mathsf{lvk}_i = (\mathrm{X}_i, \mathrm{Y}_i)$ and $\mathsf{tvk}_{i,j} = \mathrm{Z}_{i,j}$, a message set $\mathbb{M} = \{m_i\}_{i=1}^{\ell}$, a time-dependent function evaluation $F(\mathsf{ctx}, j)$, and an aggregate signature $\sigma_{\mathsf{agg},j} = (h', s)$, outputs 1 if $h' \neq 1_{\mathbb{G}}$ and the following equation holds:

$$e\left(h', \prod_{i=1}^{\ell} \mathrm{X}_i \cdot \mathrm{Y}_i^{m_i}\right) \cdot e\left((h^{\delta})^{F(\mathsf{ctx},j)}, \prod_{i=1}^{\ell} \mathrm{Z}_{i,j}\right) = e(s, v)$$

$\mathsf{RndSigTag}(\mathsf{avk}_j, \mathsf{tg}, \sigma_{\mathsf{agg},j}, r) \to (\sigma'_{\mathsf{agg},j}, \mathsf{tg}')$: Under a given aggregated verification key avk_j and with a tag tg, an aggregate signature $\sigma_{\mathsf{agg},j}$, and a random value r, the algorithm simultaneously randomizes the signature and tag as:

$$\sigma'_{\mathsf{agg},j} = ((h')^r, s^r), \quad \mathsf{tg}' = ((h^{\gamma})^r, (h^{\delta})^r)$$

Correctness. We refer readers to the full paper [53, App. C.2] for correctness analysis.

Unforgeability. The formal security proof of unforgeability for our EB-PS scheme is presented in the full version [53, App. D.2].

Theorem 4.1. The EB-PS scheme is EUF-eCMA secure under the STB-GPS assumption in the random oracle model.

5 Multi-Authority Anonymous Credentials with Epoch-Based Weights

In this section, we present Multi-Authority Anonymous Credentials with Epoch-Based Weights (MA-ACEW), the first AC scheme supporting weighted trust across distributed authorities. Existing multi-authority schemes [45, 56, 63] treat issuers uniformly, lacking the ability to capture varying trust levels. The weighted threshold signature (WTS) scheme of Das et al. [29] suggests a direction but depends on BLS signatures, making it unsuitable for standard ACs and dynamic weight updates. MA-ACEW overcomes this by integrating a modified WTS with our EB-PS framework.

5.1 Formal Definition

Suppose there are n credential issuers CI_i $(i \in [n])$. In the first epoch, each CI_i generates long-term keys $(\mathsf{lsk}_i, \mathsf{lvk}_i)$ via KGen and epoch-specific keys $(\mathsf{tsk}_{i,1}, \mathsf{tvk}_{i,1})$ via TKGen, together with an initial weight vector $\boldsymbol{w}_1 = (w_{1,1}, \ldots, w_{n,1})$. In epoch j, users interacting with CI_i obtain a long-term credential $\mathsf{cred}_{\mathsf{lt},i}$ signed under lsk_i and an epoch-specific credential $\mathsf{cred}_{\mathsf{ep},i,j}$ signed under $\mathsf{tsk}_{i,j}$.

At transition $j \to j+1$, the weight vector updates to $\boldsymbol{w}_{j+1}$ (based on external factors), each issuer erases $(\mathsf{tsk}_{i,j}, \mathsf{tvk}_{i,j})$,

and generates $(\mathsf{tsk}_{i,j+1}, \mathsf{tvk}_{i,j+1})$. Using the new key, CI_i non-interactively derives fresh epoch-specific credentials $\mathsf{cred}_{\mathsf{ep},i,j+1}$ for all users with valid long-term credentials, ensuring uniqueness via tag tg. In the context of credentials, we treat messages as attributes, while retaining the original notation m.

Definition 3. ***(MA-ACEW).*** *A Multi-Authority Anonymous Credentials with Epoch-Based Weights (MA-ACEW) is defined by the following algorithms/protocols:*

Setup: On input a security parameter λ, output public parameter pp.

IniEpKGen: Initialize the system by generating epoch-1 state and key materials for each issuer $\{\mathsf{CI}_i\}_{i\in[n]}$, including long-term key pair $(\mathsf{lsk}_i, \mathsf{lvk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$, epoch-specific key pair $(\mathsf{tsk}_{i,1}, \mathsf{tvk}_{i,1}) \leftarrow \mathsf{TKGen}(1^\lambda)$ for credential issuance, and aggregation key ak_i for inner-product argument generation. A global verification key vk is derived from all issuers' aggregation keys, along with the initial weight vector $\boldsymbol{w}_1 \in \mathbb{F}_p^n$.

UKGen: Given a message-key space S, generate a user key pair $(\mathsf{usk}, \mathsf{uvk})$ as the user's identity along with the auxiliary information aux.

Issuance: During epoch j, each user performs a one-time interaction with credential issuer CI_i to obtain a credential tuple $(\mathsf{cred}_{\mathsf{lt},i}, \mathsf{cred}_{\mathsf{ep},i,j})$ consisting of a long-term credential $\mathsf{cred}_{\mathsf{lt},i}$ and an epoch-specific credential $\mathsf{cred}_{i,j}$, where:

$$\langle \mathsf{CredObtain}(\mathsf{tg}, \mathsf{aux}, \mathbb{M}) \leftrightarrow \mathsf{CredIssue}(j, \mathsf{lsk}_i, \mathsf{tsk}_{i,j}) \rangle \rightarrow \mathsf{cred}_{\mathsf{lt},i}, \mathsf{cred}_{\mathsf{ep},i,j}$$

EpUpdate: To initiate epoch $j+1$, the system first updates the weight vector from $\boldsymbol{w}_j$ to $\boldsymbol{w}_{j+1}$. Then, each CI_i generates a new, epoch-specific key pair $(\mathsf{tsk}_{i,j+1}, \mathsf{tvk}_{i,j+1})$. Subsequently, for every user holding a valid long-term credential $\mathsf{cred}_{\mathsf{lt}}$, CI_i issues a fresh epoch credential $\mathsf{cred}_{\mathsf{ep},i,j+1}$, computed as follows:

$$\mathsf{CredIssueEp}(\mathsf{tsk}_{i,j+1}, \mathsf{tg}, j+1) \rightarrow \mathsf{cred}_{\mathsf{ep},i,j+1}$$

Gen-Policies : For clarity, we first introduce our protocol within a simplified framework. In this model, the verifier's policy is constrained to two components: the current epoch $j+1$ and a minimum weight threshold $\mathrm{W}_{\mathsf{acc}}$, denoted as $\mathsf{pol} = (j+1, \mathrm{W}_{\mathsf{acc}})$. To satisfy this policy, the user must present a set of certified attributes $\mathbb{M}$ whose cumulative weight meets or exceeds $\mathrm{W}_{\mathsf{acc}}$ for epoch $j+1$. Our construction readily extends to support a more expressive policy framework, enabling verifiers to specify arbitrary attribute disclosure requirements and complex predicates over hidden attributes. We defer the full details of this extension to Section 5.5.

Show: To satisfy a verifier's policy $\mathsf{pol} = (j+1, \mathrm{W}_{\mathsf{acc}})$, a user holding an aggregated credential $\mathsf{cred}_{\mathsf{agg},j+1}$ and a corresponding tag tg engages in an interactive protocol with the verifier. Here, tg serves as the user's pseudonymous identity. The interaction involves the user providing a zero-knowledge proof π and disclosing a set of certified attributes $\mathbb{M} = \{m_k\}_{k\in[\ell]}$. The proof π demonstrates that the credential's aggregated weight $\mathrm{W}_{\mathsf{claim}}$ satisfies the policy threshold, i.e., $\mathrm{W}_{\mathsf{claim}} \geq \mathrm{W}_{\mathsf{acc}}$.

$$\left\langle \begin{array}{l} \mathsf{CredShow}(\mathsf{tg}, \mathsf{cred}_{\mathsf{agg},j+1}, \mathbb{M}, \pi) \leftrightarrow \\ \mathsf{CredVerify}(\mathsf{pp}, \mathbb{M}, \mathsf{pol}, \{\mathsf{lvk}_i, \mathsf{tvk}_{i,j+1}\}_{i\in[\ell]}, \pi) \end{array} \right\rangle \rightarrow \{0,1\}$$

Here, we assume the full set $\mathbb{M}$ is disclosed for simplicity. In Section 5.5, we extend our construction to support flexible selective disclosure, which is achieved by partitioning the attribute set into disclosed ($\mathbb{M}_{\mathcal{D}}$) and hidden ($\mathbb{M}_{\mathcal{H}}$) subsets. As a final simplification for this presentation, we also denote the time function as j, using only the epoch index.

5.2 Secuirty Definition

We formalize security in a game-based model adapted from prior work [38, 45, 56]. To capture the dynamics of epoch-based weights, we extend this model with three new oracles: $\mathcal{O}^{\mathsf{UpdEp}}$, $\mathcal{O}^{\mathsf{ObtIssEp}}$, and $\mathcal{O}^{\mathsf{IssEp}}$. The full formalization, detailing the game mechanics and oracle interactions, is deferred to the full version [53, App. E].

Correctness. The correctness ensures that a credential showing for a non-empty attribute subset $\mathbb{M}$ always verifies successfully when the credential was honestly issued for an attribute set $\{m_i\}_{i\in[\ell]}$, and the aggregate weight of the attributes in $\mathbb{M}$ satisfies the threshold specified in the policy pol. The aggregated credential must be newly issued or properly updated within the current epoch.

Unforgeability. The unforgeability asserts that no adversary $\mathcal{A}$ can produce a valid aggregated credential $\mathsf{cred}_{\mathsf{agg},j}$ with non-negligible probability, without possessing the required credentials from the set of accepted issuers $\mathsf{CI} = \{\mathsf{lvk}_i, \mathsf{tvk}_{i,j}\}_{i\in[n]}$. The credential must show for a policy pol and a set of disclosed attributes $\mathbb{M}$. Here, $\mathcal{A}$ can obtain $(\mathsf{lvk}_i, \mathsf{tvk}_{i,j}) \in \mathsf{CI}$ through the oracles $\mathcal{O}^{\mathsf{HCI}}(i)$ and $\mathcal{O}^{\mathsf{CCI}}(i)$. Furthermore, all credentials used in the aggregation must be updated to the current epoch to ensure the validity of the showing.

Anonymity. The anonymity ensures that no adversary $\mathcal{A}$, acting as a malicious verifier, can distinguish between two users with non-negligible advantage. Furthermore, different showings of the same credential should be computationally unlinkable. In the security game, the adversary has adaptive access to an oracle that, on the input of two distinct user indexes id_0 and id_1, acts as one of the two credential owners (depending on bit b) in the verification.

Blindness. The blindness prevents the issuer from learning which user receives which credential, thus preserving unlinkability between issuance and presentation. Formally, it guarantees that a malicious issuer, $\mathcal{A}$, cannot distinguish between

two honest users during the issuance protocol, thereby severing the link between a credential's issuance and its subsequent presentation.

Definition 4 (Unforgeability)**.** *The unforgeability is defined by the security game in Fig.1. MA-ACEW is unforgeable if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon(\lambda)$ satisfies:*

$$\mathrm{Adv}^{\mathsf{EU\text{-}CMA}}_{\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{UNF}}_{\mathsf{MA\text{-}ACEW},\mathcal{A}}(\lambda) = 1]\right| \leq \epsilon(\lambda)$$

$\mathsf{Exp}^{\mathsf{UNF}}_{\mathsf{MA\text{-}ACEW},\mathcal{A}}(\lambda)$:

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \mathsf{CI} \leftarrow \mathcal{A}^{\langle \mathcal{O}^{\mathsf{CCI}}, \mathcal{O}^{\mathsf{HCI}} \rangle}(\mathsf{pp})$;
2. $\{\mathsf{lsk}', \mathsf{lvk}'\} \leftarrow \mathsf{KGen}(\mathsf{pp}); \{\mathsf{tsk}', \mathsf{tvk}'_j\} \leftarrow \mathsf{TKGen}(\mathsf{pp})$;
3. $\mathsf{CI}' \leftarrow \mathsf{CI} \bigcup \mathsf{lvk}'; \mathsf{pol} \leftarrow \mathsf{GenPolicy}(\mathsf{CI}')$;
4. $(\mathsf{cred}'_{\mathsf{lt}}, \mathsf{cred}'_{\mathsf{ep},j}) \leftarrow \mathcal{A}^{\langle \mathcal{O}^{\mathsf{User}}, \mathcal{O}^{\mathsf{ObtIss}}, \mathcal{O}^{\mathsf{Issue}} \rangle}(\mathsf{pol}, \mathsf{lvk}', \mathsf{tvk}'_j)$
5. $\mathsf{cred}'_{\mathsf{ep},j+1} \leftarrow \mathcal{A}^{\langle \mathcal{O}^{\mathsf{UpdEp}}, \mathcal{O}^{\mathsf{ObtIssEp}}, \mathcal{O}^{\mathsf{IssEp}} \rangle}(\mathsf{pol}, \mathsf{tvk}'_j)$
6. $b \leftarrow \langle \mathcal{A}^{\mathcal{O}^{\mathsf{CredShow}}} \leftrightarrow \mathsf{CredVerify}(\mathsf{pol}, \mathsf{avk}, \mathbb{M}', \mathsf{cred}'_{\mathsf{agg},j+1}) \rangle$

7. If $\forall id \in \mathcal{CU}, \exists i' \in [\ell] : \mathsf{lvk}_{i'} = \mathsf{lvk}' \wedge \{\mathsf{lvk}_i\}_{i\in[\ell]} \subset \mathsf{CI}' \wedge \mathbb{M}' \not\subset \bigcup_{i\in[\ell]} \{m_i\}, b = 1$, return 1. Otherwise, return 0.

Figure 1: Unforgeability Security Game

Definition 5 (Anonymity)**.** *The anonymity is defined by the security game in Fig.2. MA-ACEW is anonymous if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon(\lambda)$ satisfies:*

$$\mathrm{Adv}^{\mathsf{ANO}\text{-}b}_{\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{ANO}\text{-}b}_{\mathsf{MA\text{-}ACEW},\mathcal{A}}(\lambda)] - \frac{1}{2}\right| \leq \epsilon(\lambda)$$

$\mathsf{Exp}^{\mathsf{ANO}\text{-}b}_{\mathsf{MA\text{-}ACEW},\mathcal{A}}(\lambda)$

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); b \in \{0,1\}$;
2. $\langle \mathcal{O} \rangle \leftarrow \langle \mathcal{O}^{\mathsf{HCI}}, \mathcal{O}^{\mathsf{CCI}}, \mathcal{O}^{\mathsf{User}} \rangle$;
3. $(id_0, id_1, \mathsf{pol}, \{\mathsf{lvk}_i, \mathsf{tvk}_{i,j}\}_{i\in[\ell]}) \leftarrow \mathcal{A}^{\langle \mathcal{O} \rangle}(\mathsf{pp})$;
4. $\langle \mathcal{O} \rangle = \{\mathcal{O}^{\mathsf{ObtIss}}, \mathcal{O}^{\mathsf{Obtain}}, \mathcal{O}^{\mathsf{UpdEp}}, \mathcal{O}^{\mathsf{IssEp}}\}$
5. $b' \leftarrow \mathcal{A}^{\langle \mathcal{O}, \mathcal{O}^{\mathsf{Anch}\text{-}b} \rangle}(\mathsf{st})$; If $b' = b$; Otherwise, return 0.

Figure 2: Anonymity Security Game

Definition 6 (Blindness)**.** *The blindness is defined by the security game in Fig.3. MA-ACEW is blind if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon(\lambda)$ satisfies:*

$$\mathrm{Adv}^{\mathsf{BLI}\text{-}b}_{\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{BLI}\text{-}b}_{\mathsf{MA\text{-}ACEW},\mathcal{A}}(\lambda)] - \frac{1}{2}\right| \leq \epsilon(\lambda)$$

$\mathsf{Exp}^{\mathsf{BLI}\text{-}b}_{\mathsf{MA\text{-}ACEW},\mathcal{A}}(\lambda)$

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), b \in \{0,1\}$;
2. $\langle \mathcal{O} \rangle \leftarrow \langle \mathcal{O}^{\mathsf{HCI}}, \mathcal{O}^{\mathsf{CCI}}, \mathcal{O}^{\mathsf{User}} \rangle$;
3. $(id, \mathsf{pol}, \{\mathsf{lvk}_i, \mathsf{tvk}_{i,j}\}_{i\in[\ell]}, \mathbb{M}_b) \leftarrow \mathcal{A}^{\langle \mathcal{O} \rangle}(\mathsf{pp})\rangle$;
4. $\langle \mathcal{O} \rangle \leftarrow \langle \mathcal{O}^{\mathsf{ObtIss}}, \mathcal{O}^{\mathsf{Obtain}} \rangle$;
5. $b' \leftarrow \mathcal{A}^{\langle \mathcal{O}, \mathcal{O}^{\mathsf{Blch}\text{-}b} \rangle}(\mathsf{st})$; If $b' = b$, return 1, else return 0.

Figure 3: Blindness Security Game

5.3 Building Blocks for MA-ACEW

Before presenting our MA-ACEW construction, we recall the weighted threshold signature (WTS) scheme of Das et al. [29], which ensures that aggregated credential weights in each epoch satisfy the policy. The scheme relies on the polynomial identity from [7], widely used in succinct SNARK constructions:

$$x(t)b(t) = q(t) \cdot z_H(t) + t \cdot r(t) + \langle \boldsymbol{x}, \boldsymbol{b} \rangle \cdot n^{-1}$$

We adapt WTS to our framework with two changes: (i) moving from symmetric to asymmetric pairings, and (ii) splitting the Combine phase into CombPk and CombWt, which enables WTS integration. For details, please refer to the full paper [53, App. B.2].

5.4 MA-ACEW construction

In this subsection, we present the concrete construction of MA-ACEW. For simplicity, we denote the EB-PS scheme as Ψ_1 and the WTS scheme as Ψ_2, and we instantiate the time-independent function $F(\mathsf{ctx}, j)$ with access only to j in MA-ACEW. These notations and the function are consistently used in Fig. 4 and throughout this subsection.

Interactive issuing. To prevent issuers from learning user attributes, we build upon techniques from [56, 63]. For clarity, we refer to the GenAuxTag algorithm as GenUserTag and detail the credential issuance protocol. In the original EB-PS scheme, the GenAuxTag algorithm exposed plaintext attributes within its output: $\mathsf{aux} = u^\gamma \parallel u^\delta \parallel \{(m_i, \mathsf{lvk}_i)\}_{i\in[\ell]}$.

Our core modification is to replace these plaintext attributes $\{m_i\}$ with their ElGamal encryptions. Concretely, the user first generates an ElGamal key pair $(\mathsf{esk}, \mathsf{evk}) = (d, \zeta = u^d)$. For each attribute m_i, the user then computes its ciphertext $c_i = \mathsf{Enc}((h')^{m_i}, \mathsf{evk}) = (u^{k_i}, \zeta^{k_i} \cdot (h')^{m_i})$ and, in parallel, a commitment to the attribute $c_{m_i} = u^{m_i} h_1^{o_i}$. The values (d, o_i, k_i) are sampled uniformly at random from $\mathbb{F}_p$, and $h_1 \in \mathbb{G}_1$. Additionally, the user needs to generate a corresponding proof:

$$\pi_{\mathsf{CI}_i} = \mathsf{ZKPoK}\{(d, m_i, o_i, k_i) : \zeta = u^d \wedge c_{m_i} = u^{m_i} h_1^{o_i} \wedge c_i = (u^{k_i}, \zeta^{k_i} \cdot (h')^{m_i})\}$$

Setup Phase: Run $\mathsf{pp}_{\Psi_1} \leftarrow \Psi_1.\mathsf{Setup}(1^\lambda, \mathrm{T}) \wedge \mathsf{pp}_{\Psi_2} \leftarrow \Psi_2.\mathsf{Setup}(1^\lambda)$, output $\mathsf{pp} = (\mathsf{pp}_{\Psi_1}, \mathsf{pp}_{\Psi_2}) = (\mathsf{BG}, \mathsf{CRS}, \overrightarrow{v}_{\mathcal{L}}, \overrightarrow{\alpha}_{\mathcal{L}}, \overrightarrow{\beta}, \alpha, \theta, v^\eta)$.
Initial Epoch IKGen Phase: Initialize the epoch state by setting the current epoch $j := 1$.

- IKG.1 Generate the long-term keys for each issuer $\{(\mathsf{lsk}_i, \mathsf{lvk}_i)\}_{i\in[n]} \leftarrow \Psi_1.\mathsf{KGen}(\mathsf{pp}_{\Psi_1})$, where $\mathsf{lsk}_i = (x_i, y_i), \mathsf{lvk}_i = (\mathrm{X}_i, \mathrm{Y}_i)$.
- IKG.2 Set initial weight vector $\boldsymbol{w}_1 := [w_{1,1}, \ldots, w_{n,1}]$, where $w_{i,1}$ corresponds to CI_i's public key X_i.
- IKG.3 Generate the epoch-specific keys for CI_i: $\{\mathsf{tsk}_{i,1}, \mathsf{tvk}_{i,1}\}_{i\in[n]} \leftarrow \Psi_1.\mathsf{TKGen}(\mathsf{pp}_{\Psi_1}, n, 1)$, where $\mathsf{tsk}_{i,1} = z_i$, $\mathsf{tvk}_{i,1} = \mathrm{Z}_{i,1}$.
- IKG.4 Compute the verification and aggregated keys from $(\mathsf{ak}_i, \mathsf{vk}) \leftarrow \Psi_2.\mathsf{KGen}(\mathsf{pp}_{\Psi_2}, n, \boldsymbol{w}_1)$, where

$$\mathsf{vk} = (v, \alpha, \beta, v^{x(\tau)}, v^{w_1(\tau)}, v^\tau, \alpha^\tau, u^{z_H(\tau)}), \quad \mathsf{ak}_i = (v^{x_i}, v_i^{x_i}, \alpha_i^{x_i}, \theta^{x_i}, v^{\eta x_i}, \{\beta_k^{x_i}\}_{k\in[n]})$$

// Note that $v^{w_1(\tau)}$ is only utilized in epoch $j = 1$, and ak_i is to assist user to persists across all epochs.

UKGen phase: Run $(\mathsf{aux}, \mathsf{tg}) \leftarrow \Psi_1.\mathsf{GenAuxTag}(\mathrm{S})$, return $(\mathsf{usk} = (\gamma, \delta), \mathsf{uvk} = (\Gamma, \Delta), \mathsf{aux} = (u^\gamma \parallel u^\delta \parallel \{c_{m_i}, c_i, \mathsf{lvk}_i\}_{i\in[\ell]}))$ to user.
Gen-Policies Phase: The verifier sets threshold $\mathrm{W}_{\mathsf{acc}}$ and only accepts credentials of current epoch j. Return $\mathsf{pol} = (\mathrm{W}_{\mathsf{acc}}, j)$.

Issuance Phase: The user interacts with CI_i to obtain partial credentials at epoch j:

- IS.1 The user sends $(\mathsf{tg}, \mathsf{aux}, \pi_{\mathsf{tg}}, \pi_{\mathsf{CI}_i})$ to an issuer CI_i, where defined as:

$$\mathsf{aux} = (u^\gamma \parallel u^\delta \parallel \{c_{m_i}, c_i, \mathsf{lvk}_i\}_{i\in[\ell]}), \quad \pi_{\mathsf{tg}} = \mathsf{ZKPoK}\{(\gamma, \delta) : \mathsf{tg}_1 = h^\gamma \wedge \mathsf{tg}_2 = h^\delta \wedge \Gamma = u^\gamma \wedge \Delta = u^\delta\},$$
$$\pi_{\mathsf{CI}_i} = \mathsf{ZKPoK}\{(d, m_i, o_i, k_i) : \zeta = u^d \wedge c_{m_i} = u^{m_i} h_1^{o_i} \wedge c_i = (u^{k_i}, \zeta^{k_i} \cdot (h')^{m_i})\}$$

- IS.2 CI_i verifies $\pi_{\mathsf{CI}i}$ and π_{tg}, checks the format of aux, and if all pass, then computes $(\widehat{\mathsf{cred}_{\mathsf{lt},i}}, \mathsf{cred}_{\mathsf{ep},i,j})$ for the user, where:

$$\widehat{\mathsf{cred}_{\mathsf{lt},i}} = (h', c_{i,1}^{y_i}, (h')^{x_i} c_{i,2}^{y_i}), \quad \mathsf{cred}_{\mathsf{ep},i,j} = (h^\delta)^{j \cdot z_{i,j}}$$

- IS.3 The user unblinds the long-term credential to obtain $\mathsf{cred}_{\mathsf{lt},i} = (h^\gamma, (h^\gamma)^{x_i + m_i y_i})$ and stores $(\mathsf{cred}_{\mathsf{lt},i}, \mathsf{cred}_{\mathsf{ep},i,j})$.
  // Note that the user only interacts with CI_i once, as $\mathsf{cred}_{\mathsf{lt},i}$ remains valid across all epochs.

Epoch Update: When the system transitions from epoch j to epoch $j+1$, the following operations are performed:

- EP.1 Securely erase CI_i's previous $(\mathsf{tsk}_{i,j}, \mathsf{tvk}_{i,j})$, and run $\{\mathsf{tsk}_{i,j+1}, \mathsf{tvk}_{i,j+1}\}_{i\in[n]} \leftarrow \Psi_1.\mathsf{TKGen}(\mathsf{pp}, n, j+1)$ for epoch $j+1$.
- EP.2 Reset the epoch weight vector $\boldsymbol{w}_{j+1}$ and recompute the weight verification key $v^{w_{j+1}(\tau)}$, which is part of vk.
- EP.3 Each issuer CI_i computes $\mathsf{cred}_{\mathsf{ep},i,j+1} = (h^\delta)^{(j+1)\cdot z_{i,j+1}}$ for its credentialed users as part of the credential update.
- EP.4 The verifier updates pol to $(\mathrm{W}'_{\mathsf{acc}}, j+1)$.
  // Upon receiving $\mathsf{cred}_{\mathsf{ep},i,j}$ from each CI_i, the user replaces the previous epoch signature $\mathsf{cred}_{\mathsf{ep},i,j}$ with $\mathsf{cred}_{\mathsf{ep},i,j+1}$.

Show phase: The user interacts with the verifier to show a credential in epoch $j+1$:

- SH.1 The user computes the combined credential $\mathsf{cred}_{\mathsf{agg},j+1}$ with disclose set $\mathbb{M}$ in current epoch, where:

$$\mathsf{cred}_{\mathsf{agg},\mathsf{lt}} \leftarrow \Psi_1.\mathsf{AggSigLt}(\mathsf{tg}, \{(\mathsf{lvk}_i, m_i, \mathsf{cred}_{\mathsf{lt},i})\}_{i=1}^\ell), \quad \mathsf{cred}_{\mathsf{agg},\mathsf{ep},j+1} \leftarrow \Psi_1.\mathsf{AggSigEp}(j+1, \mathsf{tg}, \{\mathsf{tvk}_{i,j+1}, \sigma_{\mathsf{ep},i,j+1}\}_{i=1}^\ell),$$
$$\mathsf{cred}_{\mathsf{agg},j+1} \leftarrow \Psi_1.\mathsf{CombAggSig}(j+1, \mathsf{tg}, \mathsf{cred}_{\mathsf{agg},\mathsf{lt}}, \mathsf{cred}_{\mathsf{agg},\mathsf{ep},j+1})$$

- SH.2 According to the disclosed set $\mathbb{M}$, set the bit vector $b[i] = 1$ for $i \in \mathbb{M}$, and compute the proof as:

$$(\pi_b, \pi_{\mathsf{IPA},\mathsf{pk}}, \mathrm{X}) \leftarrow \Psi_2.\mathsf{CombPk}(\boldsymbol{b}, \{\mathsf{ak}_i\}_{i\in[n]}), \quad (\pi_{\mathsf{IPA},\mathsf{wt}}, \mathrm{W}_{\mathsf{claim}}) \leftarrow \Psi_2.\mathsf{CombWt}(\boldsymbol{w}_{j+1}, \{\mathsf{ak}_i\}_{i\in[n]})$$

- SH.3 Run $\pi_{\mathsf{IPA}} \leftarrow \Psi_2.\mathsf{MergePf}(\pi_{\mathsf{IPA},\mathsf{pk}}, \pi_{\mathsf{IPA},\mathsf{wt}}, \mathrm{X}, \mathrm{W}_{\mathsf{claim}}), (\mathsf{cred}'_{\mathsf{agg},j+1}, \mathsf{tg}') \leftarrow \Psi_1.\mathsf{RndSigTag}(\mathsf{avk}_{j+1}, \mathsf{tg}, \mathsf{cred}_{\mathsf{agg},j+1}, r)$.
- SH.4 The user sends $(\mathsf{cred}'_{\mathsf{agg},j+1}, \mathsf{tg}', u_b, \pi_b, \pi_{\mathsf{IPA}}, \mathrm{X}, \mathrm{W}_{\mathsf{claim}}, \mathbb{M})$ to the verifier.

CredVerify Phase: The verifier checks if the randomized credential is valid and satisfies the acceptance threshold defined in pol.

- CV.1 Run $\Psi_1.\mathsf{AggVerify}(j+1, \mathsf{tg}', \mathsf{avk}_{j+1}, \mathbb{M}, \mathsf{cred}'_{\mathsf{agg},j+1})$ and $\Psi_2.\mathsf{Verify}(\pi_{\mathsf{IPA}}, \mathsf{vk}, \mathrm{X}, \mathrm{W}_{\mathsf{claim}})$.
- CV.2 Check if $\mathrm{W}'_{\mathsf{acc}} \leq \mathrm{W}_{\mathsf{claim}}$. Return 1 if all verification checks are successful; otherwise, return 0.

Figure 4: MA-ACEW Construction

The auxiliary information aux is now defined as

$$\mathsf{aux} = (u^{\gamma} \parallel u^{\delta} \parallel \{c_{m_i}, c_i, \mathsf{lvk}_i\}_{i\in[\ell]})$$

And the issuing phase between user and issuer CI_i as:

- The user transmits $(\mathsf{tg}, \mathsf{aux}, \pi_{\mathsf{CI}_i}, \pi_{\mathsf{tg}})$ to the credential issuer CI_i, where

$$\pi_{\mathsf{tg}} = \mathsf{ZKPoK}\Big\{(\gamma,\delta) : \mathsf{tg}_1 = h^{\gamma} \wedge \mathsf{tg}_2 = h^{\delta} \wedge \Gamma = u^{\gamma} \wedge \Delta = u^{\delta}\Big\}$$

  is uniformly utilized for all credential issuers.

- The issuer CI_i parses aux and verifies the validity of proofs π_{CI_i} and π_{tg}. Upon successful verification, CI_i executes blind issuance on the attribute m_i, computing:

$$\widehat{\mathsf{cred}_{\mathsf{lt},i}} = (h', c_{i,1}^{y_i}, (h')^{x_i} c_{i,2}^{y_i}), \quad \mathsf{cred}_{\mathsf{ep},i,j} = (h^{\delta})^{j\cdot z_{i,j}}$$

- The user unblinds the partial long-term credential as $\mathsf{cred}_{\mathsf{lt},i} = (h', (h')^{x_i} c_{i,2}^{y_i} (c_{i,1}^{y_i})^{-d})$

Epoch transition. In MA-ACEW, the user interacts with each credential issuer CI_i only once to obtain the long-term credential. Additionally, CI_i issues epoch-based credentials $\mathsf{cred}_{\mathsf{ep},i,j}$. To streamline this process, CI_i maintains a list of registered users, identified by their unique tags tg. For each new epoch $j+1$, CI_i computes and issues fresh epoch-based credentials $\mathsf{cred}_{\mathsf{ep},i,j+1}$ for all registered users without requiring additional interaction, which eliminates the need for subsequent interactions between users and credential issuers.

During the system initialization phase, each credential issuer CI_i generates secret keys $(\mathsf{lsk}_i = (x_i, y_i), \mathsf{tsk}_{i,1} = z_i)$, where y_i is used for signing attributes and z_i for updating (signing epoch information for each user). The value $\mathrm{X}_i = v^{x_i}$ corresponds to CI_i's initial weight $w_{i,1}$. Additionally, all issuers $\{\mathsf{CI}_i\}_{i\in[n]}$ compute aggregation keys ak_i to assist the user. Assuming a Public Key Infrastructure (PKI), users can non-interactively retrieve ak_i from each CI_i, as stated in [29]. Since the weight vector $\boldsymbol{w}_j$ is publicly known and the public key vector $\boldsymbol{pk}_x$ remains valid across all epochs, no complex operations are required for subsequent updates.

When the system transitions from epoch j to epoch $j+1$, the weight vector is reset to $\boldsymbol{w}_{j+1}$, where each $w_{i,j+1}$ continues to correspond to the issuer's long-term public key X_i. This update requires only one term in the public verification key vk to be recomputed: $v^{w_{j+1}(\tau)}$. Each credential issuer CI_i updates its epoch-specific keys by executing $(\mathsf{tsk}_{i,j+1}, \mathsf{tvk}_{i,j+1}) \leftarrow \Psi_1.\mathsf{TKGen}(\mathsf{pp}_{\Psi_1}, j+1)$. Additionally, each CI_i computes new epoch-based credentials $\mathsf{cred}_{\mathsf{ep},i,j+1} \leftarrow \Psi_1.\mathsf{SignEp}(\mathsf{tsk}_{i,j+1}, j+1, \mathsf{tg})$ for each registered user. As $w_{i,j}$ is publicly known, this computation can be performed independently of the credential issuers.

Interactive Showing. According to practical scenarios, the verifier needs to define a policy pol to determine whether to accept a user's obtained credential. Here, the verifier only accepts credentials updated in the current epoch, i.e., the credential issuer CI_i must have computed a valid epoch-specific signature for the user under tg. Additionally, the verifier can adjust the acceptance threshold $\mathrm{W}_{\mathsf{acc}}$ according to the weight distribution scenario of nodes.

The user processes the set of disclosed attributes $\mathbb{M}$ and the corresponding partial credentials $\{\mathsf{cred}_{\mathsf{lt},i}, \mathsf{cred}_{\mathsf{ep},i,j+1}\}_{i\in[\mathbb{M}]}$ as follows. First, the user aggregates the long-term credentials set and the epoch-specific credential set separately. Then, these two aggregated credentials are combined into a single credential of constant size. Notably, the user can precompute the aggregated long-term credentials. Since epoch-specific credentials are re-issued by credential issuers at the beginning of each epoch, the user only needs to aggregate them when presenting a credential during the current epoch $j+1$. It is important to note that only one attribute per lvk_i can be issued (corresponding to the secret key y_i). However, this can be extended to support multiple attributes by generating a series of lvk_i, as demonstrated in [58], [63]. We will elaborate on this extension in Section 5.5.

Additionally, the user computes the corresponding IPA proofs: $\pi_{\mathsf{IPA,pk}}$ to convince the verifier that $\langle \boldsymbol{pk}_x, \boldsymbol{b}\rangle = \mathrm{X}$, and $\pi_{\mathsf{IPA,wt}}$ to convince the verifier that $\langle \boldsymbol{w}_{j+1}, \boldsymbol{b}\rangle = \mathrm{W}_{\mathsf{claim}}$ satisfies the acceptance threshold $\mathrm{W}'_{\mathsf{acc}}$, i.e., $\mathrm{W}_{\mathsf{claim}} \geq \mathrm{W}'_{\mathsf{acc}}$. These IPA proofs are merged into a single proof by taking their random linear combination using the Fiat-Shamir heuristic [35]. Notably, the proof $\pi_{\mathsf{IPA,pk}}$ can be precomputed in the previous epoch. During epoch $j+1$, the user only needs to compute the epoch-specific aggregated credentials and the weight proof.

However, the pre-computation is only applicable when the total weight of the user's obtained credentials exceeds the verifier-defined threshold pol. If the total weight is insufficient, the user must obtain additional credentials and re-compute the corresponding proof. Nonetheless, this scenario is uncommon in practice, as system parameters typically remain relatively stable across epochs, and users generally maintain a total credential weight that exceeds the required threshold.

Given the IPA proof and the combined credential, the verifier verifies the credential's validity under epoch $j+1$ by checking that: (a) The aggregated credential $\mathsf{cred}_{\mathsf{agg},j+1}$ is a valid credential on the disclosed attribute set $\mathbb{M}$, and has been updated to the current epoch $j+1$; (b) π_b is a correct proof of commitment u_b to vector $\boldsymbol{b}$, confirming that $\boldsymbol{b}$ is a bit vector; (c) π_{IPA} is a valid IPA proof for both the aggregated public key X and that the provided $\mathrm{W}_{\mathsf{claim}}$ satisfies the defined threshold in pol.

Theorem 5.1. [UNFORGEABILITY] If the EB-PS signature scheme is unforgeable, the IPA protocol satisfies knowledge soundness, and the ZKPoK is simulation-sound extractable, then the MA-ACEW construction achieves unforgeability.

Theorem 5.2. [ANONYMITY] MA-ACEW achieves anonymity if the DDH assumption holds in $\mathbb{G}_1$ and the ZKPoK protocol is zero-knowledge.

Theorem 5.3. [BLINDNESS] MA-ACEW achieves blindness if the ElGamal encryption is IND-CPA secure and ZKPoK protocol is zero-knowledge.

The complete security proofs can be found in the full version [53, App. F].

5.5 Additional Properties

In this subsection, we extend our MA-ACEW scheme to support several advanced properties, namely *Multi-Attribute Credential Issuance*, *Issuer Hiding*, and policies supporting the *Selective Disclosure of Arbitrary Attributes*. We argue that these features can be incorporated with minor modifications to our core construction. However, providing the full specification for these extensions exceeds the page limit of the conference version. We refer to the full paper [53, App. G] for the detailed construction.

5.6 Application

In this subsection, we demonstrate the practical utility of our scheme within PoS-based ecosystems, specifically instantiating it for privacy-preserving DAO governance.

Consider a DAO voting scenario where a user acts as a delegate, aggregating voting power from multiple stakeholders (who act as credential issuers). Initially, a user holding $(\mathsf{aux}, \mathsf{tg})$ interacts with these stakeholders to accumulate the necessary credentials. Here, the public key component X_i directly correlates with the i-th stakeholder CI_i's voting weight $w_{i,j}$. To prove possession of sufficient voting power, the user employs a binary participation vector $\boldsymbol{b}$. This vector ensures that only the stakeholders contributing to the aggregated credential $\mathsf{cred}_{\mathsf{agg},j}$ are selected, meaning that only those entries where $b_i = 1$ are included in the inner products for the aggregated key $\langle \boldsymbol{pk}_x, \boldsymbol{b} \rangle$ and the total accumulated weight $\langle \boldsymbol{w}_j, \boldsymbol{b} \rangle$. Consequently, during the DAO verification phase, the user generates proofs π_{IPA} and π_b to attest that the aggregated credential validly reflects the sum of the selected stakeholders' weights, thereby meeting the governance policy pol.

To ensure governance integrity over time, our system implements a dynamic epoch mechanism where credentials require an issuer-assisted refresh for each new governance cycle. Instead of relying on static permissions, a user intending to vote obtains an epoch-based credential $\mathsf{cred}_{\mathsf{ep},i,j}$ from each stakeholder CI_i with whom they have previously interacted, in order to locally update their credential to the current state. This on-demand synchronization directly addresses the challenge of authorization freshness within the DAO: it allows the governance contract to strictly verify that a delegate's voting power is currently active for the specific epoch. Moreover, due to the randomizability of the credentials, the system upholds user unlinkability, ensuring that the validation of a user's standing does not create a traceable history of their past governance activities.

6 Performance Evaluation

We implement and evaluate our constructions in Golang, providing complete implementations of both EB-PS and MA-ACEW. All code is available in an open Zenodo repository[2]. Our implementation leverages the BLS12-381 pairing-based curve and builds upon the WTS construction from [29]. To enhance computational efficiency, we employ multi-exponentiation techniques for group elements in the aggregation process. All performance measurements were conducted on a machine equipped with an Intel Core i7-1260P CPU @2.10 GHz, 16GB RAM running Ubuntu 18.04.

6.1 Implementation Benchmarks

For all benchmarks, we selected parameters to achieve 128-bit security. We first evaluated the basic operations: a single exponentiation in the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the elliptic curve took approximately 110μs and 252μs. For storage requirements, a single compressed element in $\mathbb{G}_1$ required 48 bytes, while an element in $\mathbb{G}_2$ required 96 bytes.

Communication Overhead. We first analyze the communication overhead of EB-PS and MA-ACEW constructions during the signing/issuance phase.

Constr.	U → S/CI				U ← S/CI	
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{F}_p$	Bytes	$\mathbb{G}_1$	Bytes
EB-PS	4	n	n	$192+128n$	3	144
MA-ACEW	$4+3n$	n	0	$192+240n$	4	196

Table 1: Communication Overhead of Signing/Issuance

Table 1 presents the communication costs from the user's perspective. For auxiliary information aux, EB-PS employs $\mathsf{GenAuxTag}$ while MA-ACEW utilizes $\mathsf{GenUserTag}$. Due to the encrypted messages in MA-ACEW, its communication overhead is naturally higher. Notably, Table 1 excludes the zero-knowledge proof costs, which users can selectively generate for attributes being issued. For each issuer, these additional zero-knowledge proofs require 512 bytes for an encrypted message and 256 bytes for tag proofs.

Table 2 presents the communication overhead for showing signatures or credentials in both EB-PS and MA-ACEW schemes. Due to signature aggregation capabilities, the user-side communication costs remain constant in both constructions regardless of the number of attributes. Compared to EB-PS, MA-ACEW requires additional computation for inner

[2] https://doi.org/10.5281/zenodo.17905110

Constr.	Sig./Cred. size				PK size		
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{F}_p$	Bytes	$\mathbb{G}_1$	$\mathbb{G}_2$	Bytes
EB-PS	4	0	0	192	0	$3n$	$288n$
MA-ACEW	6	5	1	800	1	$3n+7$	$720+288n$

Table 2: Communication Overhead of Verify Phase

product arguments, resulting in higher overhead. Note that the user's tag tg is included here. For public key size, we observe a linear relationship with the number of attributes since we represent unaggregated keys. For practical storage optimization, verifiers can aggregate these public keys before verification, significantly reducing space requirements.

Timing benchmark. We evaluate the computational efficiency of both EB-PS and MA-ACEW by measuring the execution time of each algorithm/phase. For each algorithm, we report both the mean execution time and standard deviation. Our measurements use Go's built-in benchmarking framework, which adaptively determines iterations for statistical significance.

Metric	GenAuxTag	SignLt	SignEp	Verify	RndSigTag
Time (ms)	0.92	0.34	0.13	1.29	0.45
Std Dev (±)	0.38	0.02	0.01	0.21	0.06
Prims(ms): $\mathbb{G}_1$ Exp.: 0.11 | $\mathbb{G}_2$ Exp.: 0.25 | Pairing: 0.85					

Table 3: Performance Evaluation of EB-PS Scheme

Table 3 presents the performance metrics of the proposed EB-PS scheme detailed in Section 4.2, including execution time and communication overhead for key operations. Our evaluation focused on critical algorithms within the EB-PS construction. The GenAuxTag algorithm was benchmarked with 128 messages, while SignLt, SignEp, and Verify algorithms were evaluated in a single-signer scenario. Notably, the Verify algorithm demonstrates higher computational complexity due to its bilinear pairing operations.

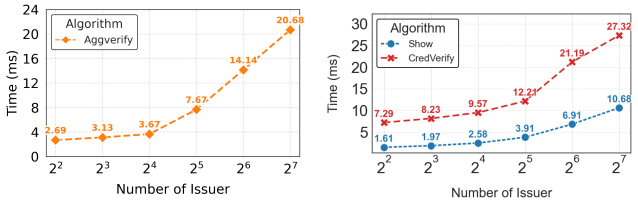
As shown in Table 4, we evaluated MA-ACEW performance across different operational phases, reporting execution times and standard deviations in a network with 64 signers. The Setup phase incurs substantial overhead due to CRS generation with complex cryptographic operations. However, the Initial IKGen phase dominates the computational cost with exceptionally high execution times, as evidenced by our measurements. This overhead stems from necessary CRS preprocessing and encompasses critical procedures: generation of long-term and epoch signing keys for all issuers, verification keys for inner product arguments, and proof generation for users. Our implementation replaces original messages in GenAuxTag with ciphertexts, increasing overhead compared to our EB-PS implementation. Furthermore, the issuance phase introduces additional computational costs due to its blind issuance mechanism, which requires users to generate ZKPoK and issuers to verify these proofs.

Phase	Time (ms)	Std Dev (±)	Notes
Setup	46.34	10.34	64 issuers
Initial IKGen	445.84	12.17	64 issuers
UKGen	5.93	0.31	10 messages
Issuance	2.74	0.86	User side
Issuance	4.55	1.52	Issuer side
Epoch Update	14.49	1.28	64 issuers

Table 4: Performance Evaluation of MA-ACEW Phases

Fig. 5a presents the computational costs of the AggVerify algorithm in EB-PS, while Fig. 5b illustrates the computational overhead of the Show and CredVerify phases in MA-ACEW. Our evaluation examines performance across varying numbers of signers/credential issuers, ranging from 4 to 128, with each issuer corresponding to a single disclosed attribute in our experimental setup.

The Show phase in MA-ACEW includes computation of Inner Product Argument (IPA) proofs. Although both schemes show increasing computational costs with more attributes, our analysis reveals that the IPA computation time grows very slowly. The observed linear growth in MA-ACEW's execution time primarily stems from the underlying EB-PS operations. Although we tested scenarios with up to 128 issuers, practical applications rarely require this many, as our MA-ACEW design allows the system to operate efficiently with fewer issuers.



(a) EB-PS performance (b) MA-ACEW performance

Figure 5: Execution time analysis of EB-PS and MA-ACEW

6.2 Smart-Contract Evaluation

We extended our evaluation to include a smart contract implementation of our MA-ACEW scheme, focusing on the issuance and credential verification functionalities. The implementation was developed in Solidity utilizing the BN254 asymmetric pairing curve. We selected BN254 due to its native support in Ethereum and superior efficiency among pairing-friendly curves in blockchain contexts.

We present our detailed performance evaluation in Table 5. Our analysis focuses on two critical phases: the Issuance phase and the credential verification phase, corresponding

Operation	Group Ops.	Gas	Est. Cost (USD)
LT. Iss.	$2\mathbb{G}_1\text{Exp} + 2\mathbb{P}$	255K	0.11
Ep. Iss.	$1\mathbb{G}_1\text{Exp}$	70K	0.03
Ver.	$19\mathbb{G}_1\text{Exp} + 15\mathbb{P}$	1077K	0.47

Table 5: Performance metrics on Ethereum

to the credential issuer and verifier operations, respectively. The Issuance phase comprises two distinct procedures: long-term credential issuance and epoch-credential issuance. The former includes computationally intensive proof verification operations, resulting in significantly higher computational costs compared to the more efficient epoch-credential issuance procedure. For the verification phase, we implement a constant-time approach that eliminates the variable complexity associated with multiple pairing operations. Since exponentiation in $\mathbb{G}_2$ incurs significant computational overhead, we pre-compute necessary group elements and optimize the protocol to process only signature pairs and inner product arguments. Our implementation leverages the batch verification technique introduced by Das et al. [29], adapting their optimized pairing-based verification framework available in the open-source codebase[3]. This approach reduces the overall verification cost by aggregating multiple pairing operations through a randomized linear combination technique, resulting in efficient and optimized pairing checks. Finally, to provide economic context for the tabulated results, USD costs were estimated using Ethereum network parameters from November 7, 2025 (ETH $\approx$ \$3,450, median gas price $\approx$ 0.128 Gwei), noting that actual costs vary with network congestion.

6.3 Comparison with Naive Solution

In this subsection, we evaluate the performance of our proposed MA-ACEW scheme through both theoretical complexity analysis and experimental implementation. We utilize a virtualized Pointcheval-Sanders (PS) signature scheme as the baseline for comparison to demonstrate the efficiency gains of our approach.

Theoretical Analysis. As shown in Table 6, the computational complexity of the baseline design depends linearly on the weight parameters. Specifically, the signing cost scales with the individual weight w_i of the i-th issuer, while the aggregation and verification costs grow linearly with the total accumulated weight W. In contrast, MA-ACEW decouples computational overhead from weight magnitude, achieving a constant signing cost per issuer and ensuring that aggregation and verification costs scale solely with the number of issuers n, irrespective of the total weight W.

Performance Benchmarking. As shown in Table 7, we fix the total accumulated weight at W $= 256$ for our evaluation.

[3] https://github.com/sourav1547/wts.

Table 6: Theoretical Complexity Comparison with Baseline

Scheme	Sign Cost *(Per Issuer)*	Agg. Cost *(Total)*	Ver. Cost *(Total)*
Baseline	$O(w_i)^*$	$O(\text{W})^\dagger$	$O(\text{W})^\dagger$
MA-ACEW	$O(1)$	$O(n)^\ddagger$	$O(n)^\ddagger$

* w_i: weight of the specific signer; † W: total weight; ‡ n: number of issuers.

Since the baseline does not support weighted issuers, it necessitates a number of issuers equal to the total weight (i.e., $n = 256$), thereby fixing its cost to this target. We first compare the worst-case scenario for our scheme, where the weight is fragmented among 256 issuers (i.e., 256 weight-1 issuers). As expected, our solution is marginally slower than the baseline in this specific setting due to the overhead of additional proofs. However, since our performance is determined by the number of issuers rather than the total weight, we gain efficiency by achieving the same weight with fewer issuers. To demonstrate this efficiency while ensuring a fair comparison (i.e., avoiding the extreme case of a single high-weight issuer), we evaluate scenarios with 128 and 64 issuers. In these more typical configurations, our scheme significantly outperforms the baseline.

Table 7: Performance Comparison with Baseline

Scheme	Showing		Verification	
	Time (ms)	Speedup	Time (ms)	Speedup
Baseline	16.1	—	31.5	—
Ours ($n = 256$)	17.3	0.93×	36.3	0.87×
Ours ($n = 128$)	10.5	**1.53**×	25.4	**1.20**×
Ours ($n = 64$)	7.0	**2.30**×	16.7	**1.89**×

Note: n denotes the number of issuers. Speedup is relative to Baseline.

7 Related Work

We focus our analysis on decentralized anonymous credentials, aggregate signatures, and PoS verification mechanisms.

Decentralized Anonymous Credentials. Garman et al. [40] introduced an innovative approach to decentralized credential systems without relying on traditional signing authorities. Their scheme leverages a public ledger (blockchain) where users register commitments to their attributes. Users prove possession of credentials by constructing zero-knowledge proofs based on RSA accumulators of ledger entries. However, this solution faces scalability challenges and may not be suitable for credentials that inherently require trusted is-

suers. To enhance security and reduce centralization in credential issuance, recent schemes have explored threshold issuance, distributing credential issuance authority among a group of entities. Such schemes require collaboration from at least a threshold number of issuers to produce a valid credential [8, 17, 32, 63]. A notable example is Coconut [63], a threshold-based anonymous credential scheme designed for blockchain environments, built upon threshold PS signatures [58]. Coconut's security is based on an interactive assumption similar to, but distinct from, the LRSW assumption [55]. Importantly, its security has been formally proven within the Universal Composability (UC) framework [59].

Multi-Authority Anonymous Credentials (MA-ACs), introduced in [45] and further studied in [56], enable the aggregation of credential showings from different issuers. The work in [56] introduces two key cryptographic primitives: tag-based aggregate signatures with randomizable tags and public keys (AtoSa) and Aggregate Mercurial Signatures with Randomizable Tags (ATMS). Notably, their scheme achieves the issuer-hiding property, which is also studied in [11, 14, 27].

While these distributed credential schemes achieve decentralized issuance, they fundamentally rely on underlying aggregate signature schemes.

Aggregate Signatures. Aggregate signatures, introduced by Boneh et al. [13], support aggregation of signatures from different parties on possibly different messages. The key requirement is succinctness of the aggregated signature. As a result of this innovative concept, several variants have been studied and developed [12, 46, 47, 57]. Among these variants, one notable example is the synchronized aggregate signature, first proposed by Gentry and Ramzan [41]. Subsequently, [2] further advanced this idea by proposing a pairing-based scheme without random oracles.

Central to our work is the Pointcheval-Sanders (PS) signature scheme [58], which provides a short signature size compared to the Camenisch-Lysyanskaya (CL) signature scheme [22]. Later, recent research has explored various structure-preserving properties in signature schemes. Ghadafi et al. [43] introduced the notion of partially structure-preserving signatures. In fully structure-preserving signature schemes [1], all messages, signatures, and public keys are group elements. Further advancing this line of research, Crites et al. [28] proposed message-indexed structure-preserving signatures. Their construction, inspired by both the PS signature scheme and Ghadafi's work [42], parameterizes messages with an indexing function, allowing aggregation of different signatures for different messages under different public keys if they share the same index.

We now move from theory to practice, exploring applications in the PoS setting, particularly unweighted multi-signature schemes and SPV-style proofs.

Unweighted Multi-signatures and SPV-style Proofs. To reduce the substantial bandwidth and storage requirements inherent in Proof-of-Stake blockchains, Drijvers et al. [33] proposed Pixel, a pairing-based forward-secure multi-signature scheme that optimizes verification costs. Following this line of work, Wei et al. [67] proposed Pixel+ and Pixel++, two forward-secure multi-signature schemes that optimize verification in PoS blockchains by aggregation of public keys. Addressing security tightness, [3] introduces a new variant of BLS multi-signatures and demonstrates how PoS protocols that currently use BLS can adopt it for fully compatible opt-in tight security. In a parallel effort to save computational resources, Baldimtsi et al. [4] propose an efficient BLS multi-signature variant for PoS blockchains that replaces per-signature randomization with a one-time public key randomization, saving significant computational resources during aggregation and verification.

Complementing signature aggregation schemes, SPV-style proofs also aim to drastically reduce verification costs. Addressing the scalability issues of traditional SPV clients, FlyClient [16] utilizes an optimal probabilistic block sampling protocol and Merkle Mountain Range (MMR) commitments to enable sub-linear light client verification. Pushing efficiency further, Vesely et al. [65] presented Plumo. Instead of downloading full headers, it securely verifies SNARK-based state transition proofs to confirm the latest network state. Furthermore, to address security in cross-chain communication, [26] proposed an accountable light client system designed to secure cross-chain bridges by validating incremental state updates and identifying misbehaving participants.

Summary of Related Work. While existing aggregate signature schemes provide elegant primitives for Anonymous Credentials, they do not directly support epoch-based validity. Moreover, previous credential schemes typically treat all issuers equally, ignoring the heterogeneity of issuer authority. In PoS settings, solutions generally rely on unweighted multi-signatures or SPV-style proofs. Unweighted multi-signatures are computationally inexpensive but misalign with the security model of PoS, as they rely on a threshold of node counts rather than the accumulated stake weight. Conversely, while recent SNARK-based light clients achieve succinct verification, they introduce significant prover-side computational overhead. Traditional SPV approaches, meanwhile, require verifiers to track a growing chain of headers and validate inclusion proofs. This introduces non-trivial bandwidth overhead and synchronization dependency. Our MA-ACEW addresses these limitations by incorporating temporal validity and flexible issuer weighting tailored for PoS-based ecosystems.

8 Conclusion

In this paper, we introduced the Epoch-Bound Pointcheval-Sanders (EB-PS) signature primitive, which enables temporal binding for signatures. Building on a concrete EB-PS construction, we developed Multi-Authority Anonymous Credentials with Epoch-based Weights (MA-ACEW), the first decentralized anonymous credential system that incorporates

weighted authority contributions. Our approach enables efficient credential updates when authority weight distributions change across epochs, addressing a significant limitation in existing systems that treat all authorities uniformly regardless of their relative trustworthiness or significance in the system.

Furthermore, we provided formal security definitions and rigorous proofs for both EB-PS and MA-ACEW constructions. To demonstrate practical viability, we implemented our schemes in Golang and developed smart contract components to evaluate the credential issuance and verification phases. Our results confirm that MA-ACEW achieves the necessary efficiency for deployment in decentralized systems such as Proof-of-Stake networks, where authority trust levels naturally fluctuate over time.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 12441101 and 62372324. We thank the anonymous reviewers and our shepherd for their constructive feedback, and Zhiqiang Ma, Gaowei Shi, and Chenyu Zhang for their insightful suggestions.

Ethical Considerations

In this section, we conduct a stakeholder analysis and discuss the potential societal impacts of our proposed weighted anonymous credential system. While our work focuses on the formal design and security analysis of a cryptographic protocol, we recognize that the deployment of such privacy-enhancing technologies requires a robust socio-technical context to be used safely.

Stakeholder Analysis. We identify four key stakeholder groups within the ecosystem of privacy-preserving proof-of-stake and governance systems. First, **Users (Credential Holders)** are the primary beneficiaries. They utilize weighted credentials to access services or participate in voting without revealing their identities, protecting them from targeting, profiling, or coercion. Their primary interest lies in the robustness of the anonymity guarantee. Second, **Service Providers and Verifiers** rely on credentials for access control or vote tallying. They benefit from the security derived from stake-backed credentials without the liability of managing user identities, though they face the operational risk of being unable to hold individual users accountable for abuse. Third, **Credential Issuers** hold the stake and issue credentials. They benefit from the *issuer-hiding* property and broader adoption driven by enhanced user privacy, but face reputational risk if credentials derived from their stake are misused. Finally, **System Operators and the Community** are concerned with the overall fairness and legitimacy of the system. While they benefit from increased participation, they could be harmed if the technology facilitates large-scale, untraceable manipulation.

Potential Impacts and Misuse Considerations. The core objective of our construction is to enable privacy-preserving authentication. However, the strong privacy guarantees also introduce potential risks. On the **positive side**, our system empowers users to act without fear of surveillance or retaliation, fostering more honest participation in sensitive contexts like voting. By guaranteeing user anonymity and issuer hiding, the system mitigates surveillance-based coercion risks. On the **negative side**, the primary ethical risk is abuse by malicious actors. Without adequate safeguards, malicious users could exploit anonymity to engage in untraceable, coordinated manipulation or launch credential-based Sybil attacks. If unchecked, this could lead to a tragedy of the commons, eroding accountability and deterring honest participants.

Mitigations and Deployment Safeguards. To address the risks associated with real-world application, we emphasize that our cryptographic protocol should not be deployed in isolation. We recommend specific architectural safeguards.

Separation of Protocol and Deployment Controls. It is essential to distinguish between the cryptographic security model and deployment-layer controls. Our security model captures the core properties of anonymous credentials, but practical defense against Sybil attacks requires external bounds on per-entity issuance. We treat issuance-rate control as a necessary measure at the deployment layer; effective deployments must impose strict bounds on credential issuance per epoch to prevent abuse.

Layered Accountability Mechanisms. Real-world systems must be designed with layered accountability mechanisms. These include **rate-limiting and economic costs** to make large-scale abuse prohibitively expensive, as well as **transparent governance and circuit-breakers**. For extreme scenarios, a decentralized process should be established to investigate and, as a last resort, respond to system-wide attacks, ensuring anonymity does not become an absolute shield for actors attempting to destroy the system itself.

Cryptographic Extensions. A natural direction for future research is to explore privacy-preserving quotas directly within the cryptographic layer. Techniques such as k-times anonymous credentials (also known as k-show) could be integrated to cryptographically enforce usage limits while maintaining user privacy, thereby bridging the gap between protocol security and operational stability.

Open Science

In adherence to the USENIX Security Symposium's open science policy, we have deposited our implementation in an open Zenodo repository (https://doi.org/10.5281/zenodo.17905110). Our repository consists of two primary components: (1) the core Golang implementation of the proposed EB-PS and MA-ACEW schemes (including cryptographic

logic and ZK-proof constructions) located in `maacew/src`; and (2) the smart contract implementation for on-chain operations located in `maacew/contracts`. Detailed instructions on code structure, dependencies, and reproduction steps are provided in the `README` file within the root directory. This ensures compliance with the conference's submission guidelines and enables transparent verification and reproduction of our results by the research community.

References

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 209–236. Springer, 2010.

[2] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. Synchronized aggregate signatures: new definitions, constructions and applications. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 473–484, 2010.

[3] Renas Bacho and Benedikt Wagner. Tightly secure non-interactive bls multi-signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 397–422. Springer, 2024.

[4] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Francois Garillot, Jonas Lindstrøm, Ben Riva, Arnab Roy, Mahdi Sedaghat, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. Subset-optimized bls multi-signature with key aggregation. In *International Conference on Financial Cryptography and Data Security*, pages 188–205. Springer, 2024.

[5] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1087–1098. ACM, 2013.

[6] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.

[7] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P Ward. Aurora: Transparent succinct arguments for r1cs. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 103–128. Springer, 2019.

[8] Kanchan Bisht, Neel Yogendra Kansagra, Reisha Ali, Mohammed Sayeed Shaik, Maria Francis, and Kotaro Kataoka. Revocable TACO: revocable threshold based anonymous credentials over blockchains. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2024, Singapore, July 1-5, 2024*, 2024.

[9] G. R. Blakley and Catherine Meadows. Security of ramp schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 242–268, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[10] Johannes Blömer, Jan Bobolz, Denis Diemert, and Fabian Eidens. Updatable anonymous credentials and applications to incentive systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1671–1685. ACM, 2019.

[11] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-hiding attribute-based credentials. In *Cryptology and Network Security: 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings 20*, pages 158–178. Springer, 2021.

[12] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 276–285, 2007.

[13] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 416–432. Springer, 2003.

[14] Daniel Bosk, Davide Frey, Mathieu Gestin, and Guillaume Piolle. Hidden issuer anonymous credential. *Proceedings on Privacy Enhancing Technologies*, 2022:571–607, 2022.

[15] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick D. McDaniel, editors, *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*, pages 132–145. ACM, 2004.

[16] Benedikt Bünz, Lucianna Kiffer, Loi Luu, and Mahdi Zamani. Flyclient: Super-light clients for cryptocurrencies. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 928–946. IEEE, 2020.

[17] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. Short threshold dynamic group signatures. In *International conference on security and cryptography for networks*, pages 401–423. Springer, 2020.

[18] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 262–288. Springer, 2015.

[19] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 21–30. ACM, 2002.

[20] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002.

[21] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[22] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual international cryptology conference*, pages 56–72. Springer, 2004.

[23] Matteo Campanelli, Anca Nitulescu, Carla Ràfols, Alexandros Zacharakis, and Arantxa Zapico. Linear-map vector commitments and their practical applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 189–219. Springer, 2022.

[24] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Springer, 1983.

[25] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[26] Oana Ciobotaru, Fatemeh Shirazi, Alistair Stewart, and Sergey Vasilyev. Accountable light client systems for proof-of-stake blockchains. *Cryptology ePrint Archive*, 2022.

[27] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In *IACR International Conference on Public-Key Cryptography*, pages 409–438. Springer, 2022.

[28] Elizabeth Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 348–382. Springer, 2023.

[29] Sourav Das, Philippe Camacho, Zhuolun Xiang, Javier Nieto, Benedikt Bünz, and Ling Ren. Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 356–370, 2023.

[30] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[31] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *Proc. Priv. Enhancing Technol.*, 2018(3):164–180, 2018.

[32] Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. Threshold bbs+ signatures for distributed anonymous credential issuance. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 773–789. IEEE, 2023.

[33] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2093–2110, 2020.

[34] e Residency Administration. e-residency: The digital identity and business residency. https://e-resident.gov.ee/, 2023.

[35] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[36] Decentralized Identity Foundation. Veramo: A javascript framework for verifiable data. https://github.com/decentralized-identity/veramo, 2024.

[37] Hyperledger Foundation. Hyperledger ursa: A shared cryptographic library for decentralized systems. https://github.com/hyperledger-archives/ursa, 2024. Accessed: 2024-12-29.

[38] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32:498–546, 2019.

[39] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. Cryptography with weights: Mpc, encryption and signatures. In *Annual International Cryptology Conference*, pages 295–327. Springer, 2023.

[40] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. *Cryptology ePrint Archive*, 2013.

[41] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*, pages 257–273. Springer, 2006.

[42] Essam Ghadafi. Short structure-preserving signatures. In *Cryptographers' Track at the RSA Conference*, pages 305–321. Springer, 2016.

[43] Essam Ghadafi. Partially structure-preserving signatures: lower bounds, constructions and more. In *International Conference on Applied Cryptography and Network Security*, pages 284–312. Springer, 2021.

[44] Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2004–2023, 2021.

[45] Chloé Hébant and David Pointcheval. Traceable constant-size multi-authority credentials. *Information and Computation*, 293:105060, 2023.

[46] Susan Hohenberger, Venkata Koppula, and Brent Waters. Universal signature aggregators. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 3–34. Springer, 2015.

[47] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 494–512. Springer, 2013.

[48] Ioanna Karantaidou, Omar Renawi, Foteini Baldimtsi, Nikolaos Kamarinakis, Jonathan Katz, and Julian Loss. Blind multisignatures for anonymous tokens with decentralized issuance. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 1508–1522. ACM, 2024.

[49] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[50] Hyoseung Kim, Youngkyung Lee, Michel Abdalla, and Jong Hwan Park. Practical dynamic group signature with efficient concurrent joins and batch verifications. *Journal of information security and applications*, 63:103003, 2021.

[51] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 308–336. Springer, 2020.

[52] Vireshwar Kumar, He Li, Noah Luther, Pranav Asokan, Jung-Min "Jerry" Park, Kaigui Bian, Martin B. H. Weiss, and Taieb Znati. Direct anonymous attestation with efficient verifier-local revocation for subscription system. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of*

the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018, pages 567–574. ACM, 2018.

[53] Chen Li, Jianting Ning, Xiulong Liu, and Yulin Liu. Differential trust: Dynamic multi-authority anonymous credentials with epoch-weighted updates. GitHub Repository: https://github.com/MA-ACEW/MA-ACEW-scheme. Full version of this paper.

[54] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 74–90. Springer, 2004.

[55] Anna Lysyanskaya, Ronald L Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99 Kingston, Ontario, Canada, August 9–10, 1999 Proceedings 6*, pages 184–199. Springer, 2000.

[56] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 30–44. ACM, 2023.

[57] Gregory Neven. Efficient sequential aggregate signed data. In *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*, pages 52–69. Springer, 2008.

[58] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29-March 4, 2016, Proceedings*, pages 111–126. Springer, 2016.

[59] Alfredo Rial and Ania M Piotrowska. Security analysis of coconut, an attribute-based credential scheme with threshold issuance. *Cryptology ePrint Archive*, 2022.

[60] Muhammad Saad, Zhan Qin, Kui Ren, DaeHun Nyang, and David Mohaisen. e-pos: Making proof-of-stake decentralized and fair. *IEEE Transactions on Parallel and Distributed Systems*, 32(8):1961–1973, 2021.

[61] Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 628–656. Springer, 2020.

[62] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[63] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. *arXiv preprint arXiv:1802.07344*, 2018.

[64] Trinsic-ID. Okapi: Collection of tools that support workflows for authentic data and identity management. https://github.com/trinsic-id/okapi, 2024. Accessed: 2024-12-29.

[65] Psi Vesely, Kobi Gurkan, Michael Straka, Ariel Gabizon, Philipp Jovanovic, Georgios Konstantopoulos, Asa Oines, Marek Olszewski, and Eran Tromer. Plumo: An ultralight blockchain client. In *International Conference on Financial Cryptography and Data Security*, pages 597–614. Springer, 2022.

[66] Shuai Wang, Wenwen Ding, Juanjuan Li, Yong Yuan, Liwei Ouyang, and Fei-Yue Wang. Decentralized autonomous organizations: Concept, model, and applications. *IEEE Transactions on Computational Social Systems*, 6(5):870–878, 2019.

[67] Jianghong Wei, Guohua Tian, Ding Wang, Fuchun Guo, Willy Susilo, and Xiaofeng Chen. Pixel+ and pixel++: Compact and efficient forward-secure multi-signatures for pos blockchain consensus. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6237–6254, 2024.

[68] Stephan Wesemeyer, Christopher J. P. Newton, Helen Treharne, Liqun Chen, Ralf Sasse, and Jorden Whitefield. Formal analysis and implementation of a TPM 2.0-based direct anonymous attestation scheme. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020*, pages 784–798. ACM, 2020.