Attribute-Based Threshold Issuance Anonymous Counting Tokens and Its Application to Sybil-Resistant Self-Sovereign Identity

Reyhaneh Rabaninejad[1], Behzad Abdolmaleki[2], Sebastian Ramacher[3], Daniel Slamanig[4], and Antonis Michalas[1,5]

[1] Tampere University, Finland
{reyhaneh.rabbaninejad,antonios.michalas}@tuni.fi
[2] University of Sheffield, UK
behzad.abdolmaleki@sheffield.ac.uk
[3] AIT Austrian Institute of Technology, Austria
sebastian.ramacher@ait.ac.at
[4] Research Institute CODE, Universität der Bundeswehr München, Germany
daniel.slamanig@unibw.de
[5] Research Institute of Sweden (RISE), Sweden

Abstract. Self-sovereign identity (SSI) systems empower users to (anonymously) establish and verify their identity when accessing both digital and real-world resources, emerging as a promising privacy-preserving solution for user-centric identity management. Recent work by Maram et al. proposes the privacy-preserving Sybil-resistant decentralized SSI system CanDID (IEEE S&P 2021). While this is an important step, notable shortcomings undermine its efficacy. The two most significant among them being the following: First, unlinkability breaks in the presence of a single malicious issuer. Second, it introduces interactiveness, as the users are required to communicate each time with issuers to collect credentials intended for use in interactions with applications. This contradicts the goal of SSI, whose aim is to give users full control over their identities. This paper first introduces the concept of publicly verifiable attribute-based threshold anonymous counting tokens (tACT). Unlike recent approaches confined to centralized settings (Benhamouda et al., ASIACRYPT 2023), tACT operates in a distributed-trust environment. Accompanied by a formal security model and a provably secure instantiation, tACT introduces a novel dimension to token issuance, which, we believe, holds independent interest. Next, the paper leverages the proposed tACT scheme to construct an efficient Sybil-resistant SSI system. This system supports various functionalities, including threshold issuance, unlinkable multi-show selective disclosure, and non-interactive, non-transferable credentials that offer constant-size credentials. The proposed construction is backed by rigorous security definitions and proofs. Finally, our benchmark results show an efficiency improvement in our construction when compared to CanDID all while accommodating a greater number of issuers and additionally reducing to a one-round protocol that can be run in parallel with all issuers.

Keywords: Anonymous counting tokens; threshold issuance; self-sovereign identity; Sybil-resistance; unlinkability

1 Introduction

Digital identity encompasses online information associated with individuals, businesses, or entities, enabling them to establish and verify their identity in both digital and real-world resources. Currently, digital identities are primarily managed through centralized and federated identity management systems. In centralized identity management, organizations store and control all digital identities within a central server. This typically involves using identifiers like email or username. Federated identity systems rely on single sign-on (SSO) and allow users to utilize a single set of login credentials (from some centralized provider), such as those provided by Facebook or Google, to access multiple applications. This server-controlled identity approach has led to a plethora of catastrophic breaches such as Yahoo's enormous user account leakage in 2013 [yah17], Facebook's data breach in 2018 [fac18] and the Microsoft breach in 2023 [ms23] to just name a few.

As a result, there has been an increasing push towards self-sovereign identity (SSI) or decentralized identity systems, leading to a paradigm shift in digital identity management. In this approach, users are liberated from the constraints of a single centralized provider and maintain complete sovereignty over how their identity information is utilized when interacting with online applications and services, thus ensuring they have the ultimate say on how their personal data is shared and used.

Self-Sovereign Identity. In this setting which is also referred to as decentralized identity (DID), users maintain a collection of credentials or decentralized identifiers and decide on their own which information should be revealed to each party. This paradigm shift has sparked numerous initiatives in recent years, with the aim of developing standards that can materialize the vision of self-sovereign identity. Prominent examples include the Decentralized Identity Foundation [dif20], the Decentralized Identifiers working group of W3C [W3C18], and the Sovrin project [Sov18]. While these initiatives have made significant strides, they have yet to address a critical requirement for many real-world systems: Sybil-resistance, or one-person-one-vote, as a fundamental aspect of fairness. In certain scenarios, such as Decentralized Finance (DeFi), the decision-making process heavily relies on collecting votes from system users or stakeholders who have made deposits within the system. Particularly in the context of Decentralized Autonomous Organizations (DAOs) [SKP+23], governance is executed through a series of proposals that members vote on via the blockchain. However, in such systems, individuals with a larger possession of governance tokens often wield more voting influence, resulting in a "one-dollar-one-vote" approach. Yet, a more equitable approach is to adopt a "one-person-one-vote" principle, which ensures equal power in decision-making for all users. To realize this aspiration, the establishment of a Sybil-resistant DID system becomes imperative. Such a system

would guarantee a single account for each individual, thereby mitigating the risks associated with identity fraud and enabling fairer participation in decentralized systems. Proof-of-personhood (PoP) [BKKJ+17] achieves Sybil-resistance by introducing pseudonym parties as periodic in-person meetings. Pseudonym parties, organized by various entities such as governments and non-profit organizations, serve as a foundation for thwarting adversaries attempting Sybil attacks by verifying the authenticity of individuals and establishing a connection between physical and virtual identities. Nevertheless, the idea's practicality is challenged by the requirement of periodic physical presence.

CanDID. Recently, Maram et al. [MMZ+21] proposed CanDID, which is a DID system achieving Sybil-resistance in a privacy-preserving way without requiring explicit provider-support. To do so, CanDID leverages oracles [ZMM+20] to securely relay the rich attested user data present in existing authoritative web servers. Technically, CanDID uses secure multi-party computation (MPC) among a committee of nodes (the so-called CanDID committee) to compute a unique value on a secret-shared deduplication attribute to achieve Sybil-resistance. This value is stored in a table and deduplicated whenever a user requests a new identity to avoid Sybil attacks. The basic design idea behind CanDID is that users can port data from legacy web accounts to create credentials and in particular a master credential (where deduplication happens). Based on this credential, users can be issued context-based credentials for specific purposes. Latter credentials inherit the Sybil-resistance property of the master credential, i.e. only one credential per context is issued.

Notwithstanding its achievements, CanDID does have certain limitations: *(I)* It achieves Sybil-resistance in an expensive way. More precisely, it utilzes MPC among committee nodes on secret shared values of the deduplication attribute which are shared with all committee nodes via an interactive protocol. *(II)* After the deduplication and master credential issuance phases are completed, CanDID requires users to communicate with committee nodes *each time* to collect context-based credentials. These credentials are intended for use in interactions with applications. This approach also contradicts the goal of self-sovereign identity, whose aim is to give users full control over their identities. *(III)* This type of structure requires frequent (complex) interactions and makes CanDID vulnerable to linking different transactions of a user by a malicious committee node, and thus reduces the provided privacy. *(IV)* Since the credentials are not linked to any private key, if the original master credential is stolen or sold on underground online markets, a malicious party can obtain context-based credentials (called application-specific credentials in our work) by presenting the stolen master credential. Consequently, the non-transferability of credentials is at risk.

Before presenting our contributions it is important to note that CanDID considers an identity system as well as a key recovery system. Our focus in this paper is solely on the former as the latter approach could be directly taken from CanDID or other recovery mechanisms adopted in Web3 [CCK+23]. Moreover, in the identity system, CanDID considers the property of accountability, i.e. the

CanDID committee can screen users to identify the credentials of suspect users (e.g. those presented on sanction lists). This is realized as a privacy-preserving fuzzy matching via MPC, can also be adopted from CanDID, and is out of the scope of this paper.

Concurrent and independent work. In a recent concurrent and independent work, Crites et al. [CKS24] present an approach that avoids interaction to introduce users to new contexts and requires no state for the issuers. The core idea is achieved via the composition of two verifiable random functions (VRF), cleverly leveraging them to ensure unlinkability while simultaneously linking by pseudonym for all signatures issued within the same context. However, Crites et al. [CKS24] only discuss the proposed Sybil-resistant "SyRA" signature for single unique attribute representing user's real-world identity. For attribute-based identification, they discuss in high-level that the context-based statement requested by the verifier can be combined with anonymous attribute-based credentials to demonstrate user attributes in a privacy-preserving way while maintaining resilience against Sybil attacks. Users will obtain keys for both SyRA signature and anonymous attribute-based credentials. This allows subsequently to engage in attribute-based identification with each transcript signed for the context requested by the verifier. Notably, the lower bound for each SyRA signature generation is 12 pairing operations *per attribute*, necessitating repetition for each context or, alternatively, interaction with issuers for each context, akin to CanDID's methodology, where users submit a set of claims required by the context to receive the per-user per-context key. They leave investigating a concrete efficient design for attribute-based signatures for future work. Our approach in contrast only needs a single per-user call to the tACT algorithm, the most resource-intensive operation, for all attributes required across different contexts in our DID framework (See Section 5 for the implementation results). We thus emphasize a more practice-oriented approach for *multiple* private attribute credentials, built on our new tACT component, which serves as a threshold token issuance mechanism.

1.1 Contributions

Our contribution in this paper is twofold:

Threshold Anonymous Counting Tokens. We introduce the notion of publicly verifiable attribute-based **t**hreshold issuance **A**nonymous **C**ounting **T**oken (tACT). Anonymous Counting Tokens (ATCs), as introduced by Benhamouda et al. in [BRS23], serve as a primitive enabling users to acquire tokens for their chosen private messages, while allowing token issuers or verifiers to impose rate limits on the number of tokens users can obtain or redeem per message. Notably, the approaches proposed in [BRS23] operate within a centralized setting, where a single entity, the issuer, handles token issuance. In contrast, we facilitate the issuance of ACT tokens in a distributed-trust environment by introducing tACT in a threshold setting. The proposed concept is accompanied by a rigorous for-

mal security model as well as an instantiation that is provably secure in this model. We believe that the concept of tACT might be of independent interest.

DID System Design. We showcase the application of the tACT primitive, by using it as a fundamental building block within our **S**ybil-resistant **S**elf-**S**overeign **Id**entity (S3ID) system, addressing important shortcomings identified in CanDID. These enhancements include:

- *Efficient Deduplication:* Our deduplication process avoids the need for resource-intensive MPC utilized by CanDID [MMZ+21]. Instead, it leverages the proposed efficient tACT executed between a user and a group of authorized issuers, each potentially representing distinct entities.
- *Non-interactive application-specific credential generation:* Our solution eliminates the necessity to communicate with the issuers to collect application-specific credentials. Here, the user locally builds application-specific credentials and directly presents them to the verifier. The proposed tACT acts as an anchor empowering users to create *user-issued* unique application-specific anonymous tags to ensure the Sybil-resistance property in direct user-application interactions.
- *Unlinkability:* In contrast to CanDID, where unlinkability breaks in the presence of a single malicious issuer, our architecture maintains unlinkability regardless of corruption of the system entities. We also present a formal definition of strong unlinkability and prove our system security under this stronger definition.
- *Non-Transferability:* The credentials issued within S3ID can only be released by the true owner.

Figure 1 outlines a system flow of our S3ID system and lists important features it achieves. S3ID can operate independently as a standalone service or seamlessly integrate with other self-sovereign identity solutions.

Evaluation. Both constructions are accompanied with implementations in Rust which is the basis for the efficency evaluation. In comparison to CanDID, we can show that our system is more efficient while at the same time supporting more issuers and higher security levels.

1.2 Technical Overview

Similar to CanDID, we consider a unique user attribute, like the Social Security Number (SSN). This is used as input to a deterministic high-entropy function and is denoted as "a".

Master credential issuance (tACT token). The user initiates the process by generating a Pedersen commitment $\mathsf{cm} = g^r \cdot u^{\mathsf{a}}$ on attribute a which serves as a blinded request to the set of N issuers, who upon reception generate partial signatures. The user then gathers partial signatures generated by authorized issuers until a threshold t out of N is reached. Then, the user generates a full unblinded

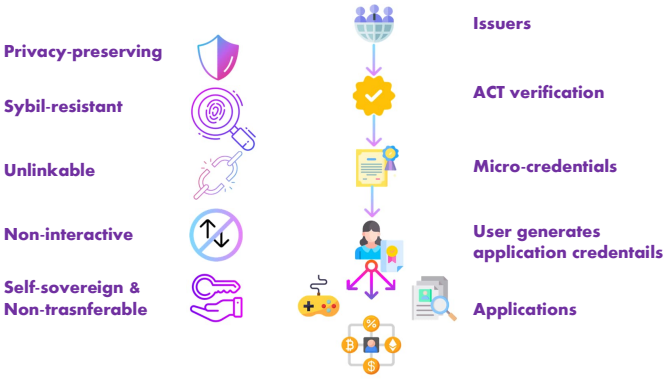

Fig. 1: S3ID system flow (right) and feature list (left).

signature by combining the collected partial signatures. This unblinded full signature effectively functions as the deterministic high-entropy representation of a (referred to as tACT), enabling issuers to perform user deduplication. However, a significant challenge arises when attempting to achieve public verifiability of tACT while maintaining the desired properties of unlinkability and anonymity of the attribute. One unsuccessful approach in achieving public verifiability involves revealing a function of randomness r used in the commitment, allowing the verifier to authenticate tACT based on the provided commitment. However, this method introduces a vulnerability where the verifier can conduct guessing attacks to identify the encapsulated attribute a, thus compromising anonymity.

Therefore, we adopt a different verification mechanism inspired by Thyagarajan et al. [TBM+20], by combining threshold secret-sharing with a *cut-and-choose* type of argument and utilizing the homomorphic property of the underlying signature. This technique allows us to prove the validity of tACT without disclosing any information regarding the enclosed attribute a. Concretely, the user secret shares the commitment cm using a threshold secret sharing scheme, where a total of n shares are generated, and at least t' shares are required to reconstruct the secret. The first $t'-1$ shares are determined as $\mathsf{cm}_j := g^{r_j} \cdot u^{\mathsf{a}_j}$ for randomly sampled values $(\mathsf{a}_j, r_j) \in \mathbb{F}_p$ and $j \in [t'-1]$. We also set $R_j = \mathsf{pk}^{r_j}$, where pk is the public key of the underlying SW signature scheme [SW13] described in Section 2.1, which is the basis of the tACT construction. The remaining shares are then generated consistently using Lagrange interpolation in the exponent, where each share with index $k \in [t', n]$ is computed based on the t'-th Lagrange polynomial basis $\ell_{t'}^{[t'-1],k}(.)$ for the points $1, \ldots, t'-1, k$ and the available $t'-1$ set of shares:

$$\mathsf{cm}_k = \left(\frac{\mathsf{cm}}{\prod_{j \in [t'-1]} \mathsf{cm}_j^{\ell_j^{[t'-1],k}(0)}} \right)^{\ell_{t'}^{[t'-1],k}(0)^{-1}}.$$

We use $\ell_j^{[t'-1],k}(.)$ as abbreviation for the j-th Lagrange basis polynomial for the points $j \in \{1, \ldots, t'-1\}$. Thereby cm_k is a valid commitment under the corresponding randomness specified below:

$$R_k = \left(\frac{\mathsf{pk}^r}{\prod_{j \in [t'-1]} R_j^{\ell_j^{[t'-1],k}(0)}} \right)^{\ell_{t'}^{[t'-1],k}(0)^{-1}}.$$

The user then sends the commitment shares $\{\mathsf{cm}_k\}_{k \in [1,n]}$ to the issuers who issue partial signatures on the blinded shares of the attribute. The user combines partial signatures to get a full signature σ_k on each cm_k and unblinds σ_k using R_k to get deterministic signature s_k on a_k.

Next, the user employs Lagrange interpolation to reconstruct valid signature s on a by selecting any set of signature shares s_k of size t'. To verify the legitimacy of s, the verifier randomly chooses a set C consisting of $t'-1$ elements. In response, the user provides the signature shares $\{s_k\}_{k \in C}$ along with the corresponding random coins $\{R_k\}_{k \in C}$ contained in the commitment shares $\{\mathsf{cm}_k\}_{k \in C}$. The verifier confirms the authenticity of s provided that both the following conditions are met:

1. All $\{s_k\}_{k \in C}$ are valid signatures on the corresponding commitment $\{\mathsf{cm}_k\}_{k \in C}$, i.e. $\forall k \in C : \mathsf{Sig.Verify}(\mathsf{pk}, \mathsf{cm}_k, R_k, s_k) = 1$
2. All commitment shares $\{\mathsf{cm}_{k'}\}_{k' \notin C}$ reconstruct to the commitment cm on attribute a, i.e. $\prod_{k_i \in C} {\mathsf{cm}_{k_i}}^{\ell_i(0)} \cdot {\mathsf{cm}_{k'}}^{\ell_{t'}(0)} = \mathsf{cm}$, with $\ell_j = \ell_j^{(k_1, \ldots, k_{t'-1}, k')}$ whereas $C = \{k_1, \ldots, k_{t'-1}\}$ considered as ordered set.

By collectively satisfying these conditions, we establish that if at least one of the partial signatures outside challenge set C aligns with its respective partial commitment, we can utilize it to reconstruct s. Consequently, a dishonest prover would be required to accurately guess the set C in advance in order to successfully pass the aforementioned verifications without s actually being a valid signature. By selecting suitable values for t' and n, one can ensure that this can only happen with negligible probability. We note that the challenge set C, can be generated non-interactively by the user implementing the Fiat-Shamir transformation [FS86]. A high-level overview of tACT construction is depicted in Figure 2.

Deduplication. Once the tACT token is built through the techniques described above, it will be used as a key ingredient in the S3ID construction. Intuitively, in the *deduplication phase* of the S3ID protocol, upon successful verification of the tACT token, the issuers proceed by comparing the token against their redeemed-token database and reject the token if this value already occurs there. Otherwise, update the database by adding redeemed token to it. Besides, as part of the registration process, the user also provides the issuer with a commitment to a uniformly chosen unique key k (represented as cm_k). Commitment

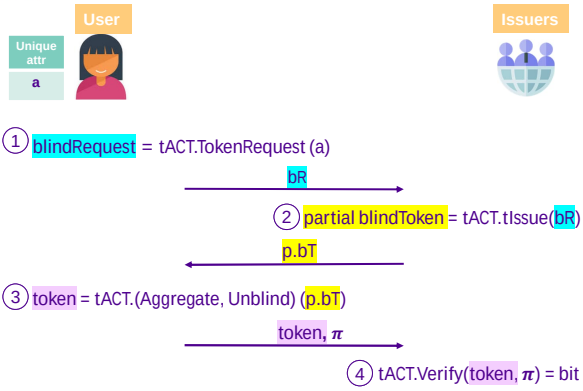

Fig. 2: tACT overview. ① User generates $\mathsf{blindRequest} = \{\mathsf{cm}, \{\mathsf{cm}_k\}_{k\in[1,n]}\}$ and sends this request to the issuers. ② Issuers generate $\mathsf{blindToken}$ on the commitment shares. ③ The user collects a threshold number of partial $\mathsf{blindToken}$, combines them to get a full signature, and unblinds the result to acquire the deterministic token. It then generates cut-and-choose proof π and sends (π, token) to the verifier. ④ The verifier checks π to ensure the validity of received token.

$\mathsf{cm_k}$ is stored alongside the user's tACT. The key k, known only to the user, facilitates meeting two essential objectives: *(I)* Non-Transferability: Micro-credentials can be released only by the true owner, *(II)* Sybil-resistance: The system achieves *one-person-one-ID* while preserving non-interactiveness and strong unlinkability – a property that will be formally defined in Section 4. In the following, we provide a detailed explanation on how these objectives are achieved.

Micro-credential issuance. After completing the deduplication phase, S3ID system enables users to acquire *micro-credentials* for their desired attributes such as age, proof of address, etc., in an anonymous manner. To obtain micro-credentials, user submits a commitment to the specific attribute. Then she proceeds to gather sufficient partial signatures from issuers and combine them to form a complete signature which is subsequently unblinded. Micro-credentials inherit the Sybil-resistance property of the tACT tokens, i.e. only one micro-credential per user attribute is issued.

Application-specific credentials[6]**.** With obtained micro-credentials, the user is empowered with a fine-grained disclosure of its attributes in the *application credentials phase*. In particular, when using an application that requires specific attributes, the user locally aggregates the relevant micro-credentials into a compact application credential and shows the aggregated result to the verifying party. The verification of application credentials simultaneously verifies all aggregated micro-credentials in a batch and thus is efficient.

The design of micro-credentials prevents malicious users from obtaining the application credential by replacing a different combination of micro-credentials

[6] We recall that CanDID calls them context-based credentials.

instead of the designated ones required by the application (*replace attack*). To counter this threat, we define a global set of admissible attributes, where each index corresponds only to a specific attribute. Issued micro-credentials are bound to these indices. Moreover, our construction guarantees that only the rightful owner can release the issued micro-credentials (*non-transferability*). This is achieved by using the following technique: issued micro-credentials are *tethered* to the committed unique user key k which prevents malicious users from obtaining the application credential by employing a combination of micro-credentials that do not belong to them. The presence of k in issued micro-credentials ensures that the user must possess k to construct tags during the application credential phase. Thus, users are compelled to build application credentials based on their genuinely owned micro-credentials. This reinforces non-transferability.

The construction so far provides the following desirable features: **(1) Threshold ACT**. The proposed tACT framework facilitates the issuance of ACT tokens in a distributed-trust environment, in contrast to the approach in [BRS23]. **(2) Uniqueness and Fast Deduplication**. User identities are deduplicated, which is highly beneficial for various applications, such as e-voting. Moreover, the deduplication process does not involve costly multi-party computation methods as offered by CanDID [MMZ+21]. **(3) Blindness and Attribute-Membership Privacy**. Throughout the entire S3ID processes, issuers are unable to obtain any information regarding user attributes. Besides, issuers are unable to learn whether there is a credential for a specific attribute value within the system. **(4) Non-Transferability**. The credentials can be released only by the true owner. **(5) Batch Verification**. The verification of application credentials simultaneously verifies all aggregated micro-credentials in a batch manner and thus is efficient.

The challenge that we still need to address is limiting the number of submissions per user per application (*Sybil-resistance*), while preserving strong unlinkability and non-interactiveness. CanDID employs an approach mitigating the Sybil attack, where a credential for an application with a unique application name is issued corresponding to the unique pseudonym of the user initiating each query. However, this method presents two significant problems: Firstly, it exhibits interactivity as it requires user engagement with a threshold number of issuers to obtain the necessary application credentials before connecting with the verifier. The interactive nature of this process can lead to system inefficiencies. Secondly, and more importantly, this method compromises the principle of unlinkability. By learning the user's unique pseudonym, issuers can establish links between different user transactions. This violation of unlinkability undermines the intended level of privacy and anonymity in the system. Therefore, despite its initial attempt to address the Sybil attack, the CanDID approach encounters two significant problems: interactivity and the compromise of unlinkability.

Adding Sybil-resistance. Our proposed system introduces a user-empowered solution to address the Sybil-resistance property by allowing users to generate their own unique application-specific anonymous tags for use in direct user-application interactions. During the redemption process, the user presents the

Table 1: Comparison of our construction over CanDID [MMZ+21] identity system.

Features	unf.	blind.	Sybil-resistance	unlink.	non-transfer.	non-interact. ‡
CanDID [MMZ+21]	✓	✓†	✓	✗	✗	✗
S3ID	✓	✓	✓	✓	✓	✓

† CanDID is pseudonymous rather than being fully anonymous; i.e., the issuer learns a unique pseudonym as the identifier of a user who initiates each query and is associated with that user's CanDID credentials.

‡ This property is with respect to the AppCred phase. Dedup and MicroCred phases are intrinsically interactive, akin to the CanDID system.

unique tag along with the corresponding application credential to the relying party. The verifier then checks if the tag for the same application message l and the same user has already been redeemed by checking its database, preventing double-submission, and subsequently examines the application credential. However, a challenge arises during this redemption check, as it is crucial to remove any link to the user's identity while retaining the ability to verify that the tag was generated by a user-registered key k. To overcome this challenge, we adopt an approach close to the work presented by Benhamouda et al. [BRS23] by employing a pseudorandom function (PRF) as the token issuance mechanism. However, in our case, the main difference is that users may compute this PRF by themselves rather than relying on an issuer. The PRF is evaluated under the user-registered key k on the *unique* application message l. The authors note that this is related to *scope-exclusive pseudonyms*, a concept employed in anonymous credential systems [CDL+13]. The adopted PRF has algebraic properties that enable it to seamlessly integrate with efficient zero-knowledge proofs based on Sigma protocols and can be made non-interactive using the Fiat-Shamir heuristic for the proof of correct evaluation. Therefore, the user efficiently demonstrates the correct evaluation of the tag on the message with respect to the key k encapsulated in all user micro-credentials, in a zero-knowledge manner. This can be achieved through two different PRF constructions: $\mathsf{PRF}(\mathsf{k}, l) = \mathsf{H}(l)^{\mathsf{k}}$, or $\mathsf{PRF}(\mathsf{k}, l) = g^{(\mathsf{k}+l)^{-1}}$, the former being the Naor-Pinkas-Reingold PRF [NPR99] and the latter being the Dodis-Yampolskiy PRF [DY05]. Since l is known by the verifier, the former PRF can also be efficiently proven in zero-knowledge. By employing this mechanism, users gain the ability to generate unique application-specific tags that can be verified with an efficient proof of correct evaluation. As a result, in addition to previously mentioned features, the system achieves the **(6) Sybil-resistance** property in a **(7) non-interactive** and **(8) unlinkable** manner. An overview of our S3ID identity system is depicted in Figure 3. We also provide a high-level comparison of our construction over CanDID in Table 1.

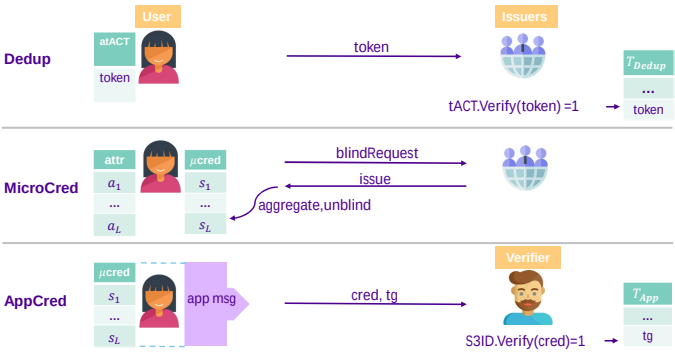

Fig. 3: S3ID overview. ① **Dedup**: User submits tACT token to the issuers, who verify the token and compare it against their database $\mathsf{T}_{\mathsf{Dedup}}$ and reject if $\mathsf{token} \in \mathsf{T}_{\mathsf{Dedup}}$. Otherwise, they update $\mathsf{T}_{\mathsf{Dedup}}$ by adding $\mathsf{token}, \mathsf{cm}_{\mathsf{k}}$ to it. ② **MicroCred**: User interacts with the issuers to obtain micro-credentials $\{\mathsf{s}_i\}_{i=1}^{L}$ on her attributes $\{\mathsf{a}_i\}_{i=1}^{L}$. ③ **AppCred**: For an application with a unique message, the user locally aggregates micro-credentials encapsulating a subset of attributes required by the application and, as a result, shows the succinct application credential value to the verifying entity. In addition, the user generates a tag under the registered key k on the unique application message to enable limiting double submissions by the verifier. The whole AppCred phase is executed locally on the user's device and is devoid of any interactions involving the issuers.

1.3 Application Domains

In general, DID systems are a key-enabling technology for today's and tomorrow's distributed world. Our lives increasingly depend on digital services, where centralized identity mechanisms are becoming an increasingly serious threat to our free society. A cryptographically secure and efficient DID mechanism that is Sybil-resistant and unlinkable and can cope with actual threats of identity theft and privacy breach, as developed within S3ID, helps to protect individual privacy and secure personal data. Here, we exemplify applications of interest that our S3ID construction could be beneficial to.

Proof of Unique Personhood. S3ID can serve as evidence of a person's distinct human identity and distinguish them from bots or other automated entities.

Passwordless Authentication. A blockchain wallet integrated with S3ID enables users to effortlessly access any portal using a single set of login credentials, representing an evolved form of single-sign-on (SSO).

Simple Know Your Customer (KYC). The DID for each user can be reused on different platforms for user onboarding purposes on various DeFi Decentralized Applications (DApps) without incurring the expenses associated with performing KYC for each individual platform.

Reputation and Credit Scoring Systems. In contrast to Web 2.0, using DID, Web 3.0 users have the ability to transfer and aggregate their reputation data across multiple DApps, without the need to make a fresh start each time.

Real-world Asset Tokenization: S3ID can be instrumental in the process of tokenizing real-world assets, ensuring a one-to-one correspondence between the physical asset itself and the token issued on-chain.

1.4 Related Work

Subsequently, we discuss approaches that are related to our new primitive.

Anonymous Credentials (AC) [PS16, San20, HS21, CLPK22] are a valuable tool for strong authentication with built-in access control and play a vital role in privacy safeguarding. In an AC system, credentials are issued by trusted issuer(s) and certify some set of user attributes. Credential holders can show that they possess a credential whose hidden attributes satisfy some predicate in a zero-knowledge manner. It is crucial to maintain the unlinkability between multiple showings of the same credential, even in situations where issuers and verifiers collude. AC technologies have been applied in diverse domains, including PrivacyPass by Cloudflare [DGS+18], Google's enhanced variant integrated into the Trust Tokens API [KLOR20], and the DIT proposal by Facebook [HIJ+21]. The Hyperledger system [C+16] facilitates the integration of anonymous credentials with blockchains by supporting CL credentials [CL04] through a trusted third-party issuer. However, such a system becomes vulnerable in situations where the issuer's behavior turns malicious. There are various proposals for threshold-issuance AC schemes [SABB+18, MSM23, DKL+23]. Said systems leverage threshold cryptography and enable multiple authorities to jointly issue credentials, hence proving well-suited for distributed ledger systems. Another line of research in AC systems is focused on attaining the multi-authority feature. This feature facilitates a compact and efficient showing of multiple credentials issued by various issuers [HP23, MBG+23]. However, it is important to highlight that a notable limitation of many current AC systems is their failure to address the critical requirement of Sybil-resistant credential issuance, a property that holds significant importance in numerous applications.

Anonymous Counting Tokens (ACT) is a new primitive introduced recently in a work by Benhamouda et al. [BRS23]. This novel primitive empowers users to acquire tokens on private messages of their preference, while enabling issuers or verifiers to enforce rate limits on the number of tokens that users can obtain or redeem for each message. An intriguing challenge that arises in the design phase of an ACT system is the apparent conflict between the desire for anonymity, which prevents mapping tokens to user identities, and the need to set limits on individual user contributions. Additionally, it is important to ensure that diverse users can acquire anonymous tokens for the same message, while keeping this information concealed from the issuer to preserve the property of unlinkability. Existing tools in the field of secure computation including Multi-Party Computation (MPC) [Gol09] and secure aggregation constructions [BIK+17, BBG+20],

address the problem of computing aggregates, while preserving privacy for individual contributions. However, these solutions fall short in providing rate limiting for user inputs, particularly in scenarios where user contributions must be generated while maintaining user anonymity. The work in [BRS23] proposes two approaches for the construction of ACTs. The first approach employs PRF evaluation as the token issuance mechanism, incorporating keyed verification that is only advantageous in scenarios where the issuer and verifier are the same entity. The second approach harnesses the concept of Equivalence Class Signatures (EQS) [FG18, FHS19] to establish an ACT scheme. However, it is worth noting that both approaches operate within a centralized setting, where a single party, known as the issuer, is responsible for issuing tokens to users.

2 Preliminaries

We recall some notions that are required for our constructions. Our notation, Pedersen commitments, zero-knowledge proofs of knowledge (ZKPoK), and oracle systems are discussed in Appendix A.

2.1 (Threshold) SW

Here, we briefly recall a variant of BLS signatures [BLS01] as introduced by Shacham and Waters in [SW13] that enables combining signatures on scalar messages.[7] We note that for simplicity we present it by hashing the message m into the group, but one can easily use any other set (e.g., an index set) if there is a one-to-one correspondence between the message space and the index set. For consistency with the original scheme [SW13] and the TS-UF-1 security result that we rely on below [BL22], we describe our scheme in the symmetric pairing setting and note that any such protocol can be (automatically) translated to the asymmetric setting [AGOT14, AGH15, AHO16]. In the following, we describe a threshold version of this signature scheme denoted as tSW.

- $\mathsf{pp} \leftarrow \mathsf{tSW.Setup}(1^\lambda)$: Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear group of prime order p, where p is a λ bit prime, and g and u are generators of $\mathbb{G}_1$. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$ be the BLS hash. The algorithm outputs system public parameters $\mathsf{pp} = (e, p, \mathbb{G}_1, \mathbb{G}_2, g, u, \mathsf{H})$.
- $(\mathsf{pk}, \{\mathsf{sk}_j\}_{j\in[1,N]}) \leftarrow \mathsf{tSW.KeyGen}(1^\lambda, t, N)$: The parties execute a distributed key generation protocol [KHG12] to generate BLS secret signing key sk where party $j \in [1, N]$ receives the share sk_j of sk such that t shares of the secret reconstruct sk. The public key is set to $\mathsf{pk} = g^{\mathsf{sk}}$.
- $\sigma_j \leftarrow \mathsf{tSW.Sign}(\mathsf{sk}_j, m)$: On input secret key sk_j and message m, outputs a partial signature $\sigma_j = (\mathsf{H}(m) \cdot u^m)^{\mathsf{sk}_j}$.
- $\sigma \leftarrow \mathsf{tSW.Comb}(\{\sigma_c\}_{c\in C})$: On input t partial signatures $\{\sigma_c\}_{c\in C}$ where $C = \{c_1, \ldots, c_t\} \subset [N]$ considered as ordered set, outputs a full signature $\sigma = \prod_{c_j \in C} \sigma_{c_j}^{\ell_j(0)}$ with ℓ_j being the j^{th} Lagrange polynomial for points $c_1, \ldots, c_t$.

[7] Note that this is not possible with plain BLS.

– $\{0,1\} \leftarrow \mathsf{tSW.Verify}(\mathsf{pk}, m, \sigma)$: On input a public key pk, message m, and signature σ checks whether $e(\sigma, g) = e(\mathsf{H}(m) \cdot u^m, \mathsf{pk})$ and outputs a bit $b \in \{0,1\}$.

Remark. Let us use an index set I instead of the message space for hashing now and i corresponds to m. We note that the $\mathsf{tSW.Sign}$ algorithm instead of signing commitment u^m can equivalently take a Pedersen commitment $\mathsf{cm} = \mathsf{Ped.Commit}(m; r) = g^r u^m$ as input[8], and output $\sigma_j = (\mathsf{H}(i) \cdot \mathsf{cm})^{\mathsf{sk}_j}$ as partial signature. In such case, we call it $\mathsf{tSW_{Ped}.Sign}$. Also, given a signature σ under $\mathsf{tSW_{Ped}.Sign}$ for commitment $\mathsf{cm} = g^r u^m$ on message m with index i, $\mathsf{tSW_{Ped}.Verify}(\mathsf{pk}, \mathsf{cm}, i, \sigma)$ checks whether $e(\sigma, g) = e(\mathsf{H}(i) \cdot \mathsf{cm}, \mathsf{pk})$. Furthermore, note that one can publicly convert between signatures of the two schemes. In particular, given a signature σ under $\mathsf{tSW_{Ped}}$ for $g^r u^m$ one can obtain signature σ' under tSW by setting $\sigma' = \sigma \cdot \mathsf{pk}^{-r}$.

As shown in [SW13], such signatures can be easily aggregated across different messages and this simplifies the verifier's task by facilitating batch verification, wherein all aggregated signatures can be simultaneously verified in a single round.

Security. As shown by Bach and Loss in [BL22], the threshold BLS signature scheme can be shown adaptively TS-UF-1 secure [BCK+22], i.e., under a strong notion where the adversary is also allowed to query partial signatures for the forged message with up to $\frac{N}{2}$ corruptions, under the k-OMDL assumption in the ROM and AGM. As we discuss later in Section 3, for our purpose static security suffices. But every scheme with adaptive corruptions is clearly also secure under static ones. We claim the following:

Theorem 1. *The tSW scheme given above is statically TS-UF-1 secure under the same assumptions as in [BL22] (Theorem 4.1).*

Proof (Proof sketch). We can turn any adversary against tSW into one against threshold BLS (tBLS). Note that the reduction can set up $u = g^v$ and pk as well as $\{pk_i\}_{[i \in N]}$ are available to the reduction. For every partial signing query for message m of the tSW adversary, we can forward m to the partial signing query of the tBLS challenger. For result σ_i we compute $\sigma_i \cdot pk_i^{vm}$ and give it to the tSW adversary. It can be seen that this is a valid partial signature for tSW. For the forgery attempt σ^*, the reduction outputs $\sigma^*(pk^{vm^*})^{-1}$ to the tBLS challenger. It can easily be checked that the simulation is perfect and any valid forgery for tSW is one for tBLS under the same winning conditions.

[8] It can also take a generalized Pedersen commitment [Ped91].

3 Attribute-Based Threshold Issuance Anonymous Counting Tokens

3.1 Definitions and Security Properties

Now, we introduce the formal definitions for attribute-based threshold anonymous counting tokens (tACTs) and present the security requirements they must meet.

Definition 1 (tACT). *An attribute-based threshold anonymous counting token (tACT) scheme comprises a set of algorithms that are defined as follows (public parameters* pp *are assumed to be an implicit input to all algorithms):*

- $(\mathsf{pp}, \{\mathsf{prv}_j\}_{j\in[1,N]}) \leftarrow \mathsf{Setup}(1^\lambda, t, N)$*: On input security parameter* 1^λ*, and parameters* t, N *representing the threshold and overall number of issuers it outputs system public parameters* pp *and private parameters*[9] prv_j *for the token issuer* $I_j, j \in [1, N]$*.*
- $(\mathsf{st}_{\mathsf{Rg}}, \mathsf{cm}) \leftarrow \mathsf{Register}(\mathsf{a})$*: On input the public parameters* pp *for the tACT scheme and unique user attribute* a*, the user outputs a commitment* $\mathsf{cm} = \mathsf{Commit}(\mathsf{a}; r)$ *and a registration state* $\mathsf{st}_{\mathsf{Rg}}$ *including* a, r*.*
- $(\mathsf{blindRequest}, \mathsf{rand}_\mathsf{a}) \leftarrow \mathsf{TokenRequest}(\mathsf{st}_{\mathsf{Rg}}, \mathsf{cm})$*: This algorithm is run by the user, on input commitment* cm *and registration state* $\mathsf{st}_{\mathsf{Rg}}$ *outputs a blinded token request denoted as* $\mathsf{blindRequest}$ *(including* cm*) and accompanying state information* rand_a *(including* $\mathsf{st}_{\mathsf{Rg}}$*).*
- $\mathsf{blindToken}_j \leftarrow \mathsf{tIssue}(\mathsf{blindRequest}, \mathsf{prv}_j)$*: Is run by issuer* I_j *holding private* prv_j*, by which the user obtains a blinded partial token embedding the private attribute* a*.*
- $\mathsf{blindToken} \leftarrow \mathsf{Aggregate}(\{\mathsf{blindToken}_j\}_{j\in[t]})$*: Is run by the user to combine any subset of* t *partial tokens into a blinded token* $\mathsf{blindToken}$*.*
- $\mathsf{token} \leftarrow \mathsf{Unblind}(\mathsf{blindToken}, \mathsf{rand}_\mathsf{a})$*: The user, given the blind token* $\mathsf{blindToken}$ *and the corresponding state information* rand_a *generated during the* $\mathsf{TokenRequest}$ *phase for attribute* a*, runs this algorithm to generate the unblinded token* token *specific to the private attribute* a*.*
- $\pi \leftarrow \mathsf{Prove}(\mathsf{token}, \mathsf{aux}, \mathsf{rand}_\mathsf{a})$*: Is run by user and inputs the token* token *embedding the private attribute* a*, auxiliary information* aux*, and its corresponding state information* rand_a*, and outputs a proof* π*.*
- $\{0, 1\} \leftarrow \mathsf{Verify}(\mathsf{token}, \pi, \mathsf{blindRequest})$*: On input the token* token*, proof* π*, and blinded token request* $\mathsf{blindRequest}$*, outputs a bit* $b = 1$ *if the verifier accepts* token *as a valid token with respect to the* $\mathsf{blindRequest}$*, and* $b = 0$ *otherwise. We note that during the verification process, the verifier remains completely unaware of any information regarding attribute* a*.*

Properties. A tACT system is called secure if it is correct, unforgeable, and unlinkable as formally defined below:

[9] This is usually run via a distributed key generation protocol.

Definition 2 (Correctness). *A tACT scheme is called correct if any token that is produced by honestly following the tACT algorithms can be successfully verified, i.e. for any sets of system and issuers' parameters* $(\mathsf{pp}, \{\mathsf{prv_j}\}_{j\in[1,N]}) \leftarrow \mathsf{Setup}(1^\lambda)$*, all attributes* a *and all state information* $\mathsf{st_{Rg}}, \mathsf{rand_a}$*,* $(\mathsf{st_{Rg}}, \mathsf{cm}) \leftarrow \mathsf{Register}(\mathsf{a})$ $(\mathsf{blindRequest}, \mathsf{rand_a}) \leftarrow \mathsf{TokenRequest}(\mathsf{st_{Rg}}, \mathsf{cm})$*,* $\mathsf{blindToken}_j \leftarrow \mathsf{tIssue}(\mathsf{blindRequest}, \mathsf{prv}_j)$*,* $\mathsf{blindToken} \leftarrow \mathsf{Aggregate}(\{\mathsf{blindToken}_j\}_{j=1}^t)$*,* $\mathsf{token} \leftarrow \mathsf{Unblind}(\mathsf{blindToken}, \mathsf{rand_a})$*,* $\pi \leftarrow \mathsf{Prove}(\mathsf{token}, \mathsf{aux}, \mathsf{rand_a})$*, the verification algorithm* $\mathsf{Verify}(\mathsf{token}, \pi, \mathsf{blindRequest})$ *outputs* 1 *with overwhelming probability.*

Adversary classes. The unforgeability definition encompasses a category of adversaries denoted by $\mathcal{A}$ who can corrupt up to $t-1$ out of N issuers. We note that while for conventional threshold signatures it might be desirable to support adaptive security for very large N [CKM23], in our applications N can be assumed to be reasonably small, i.e., having a few issuers, and thus we can obtain adaptive security for free via a simple guessing argument which incurs a $\binom{N}{t}$ loss. Additionally, $\mathcal{A}$ holds the ability to corrupt any number of users within the system. For unlinkability, $\mathcal{A}$ can corrupt *any* number of issuers and verifiers.

Unforgeability. This property ensures that an adversary controlling issuing entities in corrupted set CS cannot generate tokens for more attributes than the ones for which it has requested tokens. Furthermore, it cannot generate more than one token for a specific attribute per user.

Definition 3 (Unforgeability). *A tACT scheme is unforgeable if for any PPT adversary* $\mathcal{A}$ *and any* $T \geq 0, R \geq 0$

$$\mathbf{Adv}^{UF}_{tACT,\mathcal{A}}(\lambda) := \Pr[UF_{tACT,\mathcal{A}}(\lambda) \rightarrow 1] \leq \mathsf{negl}(\lambda),$$

where the experiment $UF_{tACT,\mathcal{A}}(\lambda)$ *is defined in Figure 4.*

Unlinkability. This property ensures that the adversary $\mathcal{A}$ who has the ability to corrupt all issuers cannot link user token requests or unblinded tokens with the private user attribute. Figure 5 formally defines the game $\mathrm{UNLINK}_{tACT,\mathcal{A}}(\lambda)$. The intuition behind this definition is as follows. The adversary $\mathcal{A}$ that has corrupted all issuers, can instruct users in the system to generate blind requests bR for attributes of its choice through the $\mathcal{O}_{\mathsf{TokenRequest}}$ oracle. The adversary should distinguish blind requests bR (oracle $\mathsf{Chl_{Issue}}$) for one of two adversary chosen attributes; or it needs to distinguish unblinded tokens generated by the $\mathcal{O}_{\mathsf{Unblind}}$ oracle. The definition requires that $\mathsf{tACT.Unblind}$ and $\mathsf{tACT.Verify}$ succeed on both the blind tokens provided by the adversary.

Definition 4 (Unlinkability). *A tACT scheme is unlinkable if for PPT adversary* $\mathcal{A}$ *the advantage defined as*

$$\mathbf{Adv}^{UNLINK}_{tACT,\mathcal{A}}(\lambda) := |2\Pr[UNLINK_{tACT,\mathcal{A}}(\lambda) \rightarrow 1] - 1|,$$

where the experiment $UNLINK_{tACT,\mathcal{A}}(\lambda)$ *is defined in Figure 5, is negligible, i.e.,*

$$\mathbf{Adv}^{UNLINK}_{tACT,\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda).$$

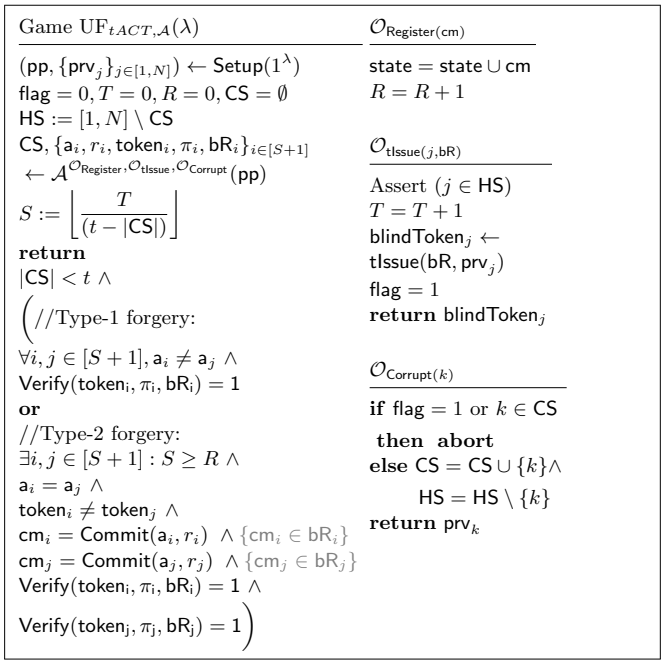

Fig. 4: Unforgeability experiment for a tACT scheme.

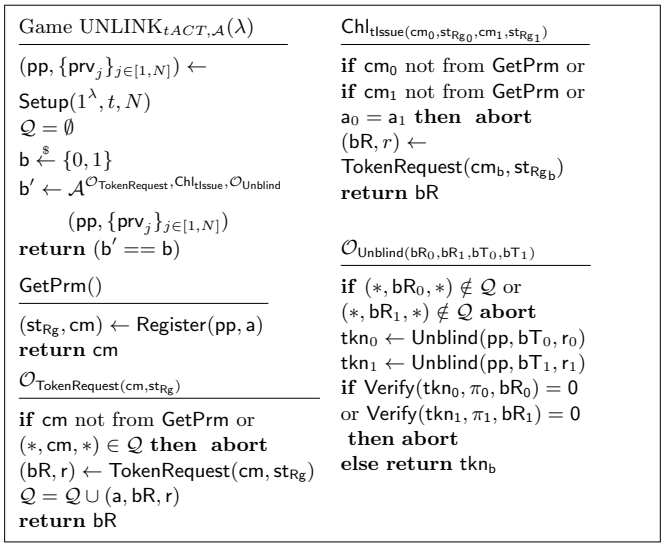

Fig. 5: Unlinkability experiment for a tACT scheme.

3.2 tACT Construction

Now we present our tACT construction. Consider a unique user attribute denoted as a, e.g., the Social Security Number (SSN). In order to obtain the token, we will need to obliviously evaluate a deterministic high-entropy function F of a in a distributed manner, where the issuers share the secret key sk, while the user has the message $(\mathsf{a}, r_{\mathsf{a}})$ as input. Upon completion of the protocol, the user is provided with the output $\mathsf{F}(\mathsf{a}, r_{\mathsf{a}})$, while the issuers gain no information. We further require $\mathsf{F}(\mathsf{a}, r_{\mathsf{a}})$ to be *anonymously* verifiable; i.e., a public verifier can verify the token without learning anything about a. The key ingredients for constructing tACT are Pedersen commitments $\mathsf{Ped} = (\mathsf{Setup}, \mathsf{Commit})$, and threshold SW signature $\mathsf{tSW}_{\mathsf{Ped}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Comb}, \mathsf{Verify})$. We also let n be a statistical security parameter and set $t' = \frac{n}{2} + 1$, with (t', n) used as parameters for the cut-and-choose. Our tACT scheme is presented in Algorithm 1. In the present scenario, tACT construction integrates the throttling property during the redemption process, which prevents a user from redeeming a token multiple times.

It begins with the issuers executing a distributed key generation protocol [KHG12] to generate shares of SW secret signing key for each issuer $j \in [1, N]$ and publish public key pk. Next, in $\mathsf{tACT.Register}$, a user generates a commitment cm to a unique attribute a. The user then secret shares the commitment using a (t', n) threshold secret sharing scheme in the $\mathsf{tACT.TokenRequest}$ algorithm. These partial commitments serve as $\mathsf{blindRequest}$, upon which the issuers generate partial $\mathsf{blindTokens}$ in the $\mathsf{tACT.tIssue}$ algorithm. The user then gathers threshold many partial signatures generated by authorized issuers. These partial signatures are then combined to form a complete signature, which is then unblinded ($\mathsf{tACT.(Aggregate, Unblind)}$). The resulting unblinded full signature is deterministic high-entropy function of a, referred to as token.

To prove the validity of token in the Prove algorithm while preserving attribute anonymity, the user non-interactively determines a set C containing $t' - 1$ elements. The user then provides the unblinded shares $\{s_k\}_{k \in C}$ along with the corresponding random coins contained in the commitment shares $\{\mathsf{cm}_k\}_{k \in C}$. The verifier confirms the authenticity of token provided that the conditions in the Verify algorithm are collectively satisfied.

For the security analysis of tACT, we refer to Appendix B.1.

Algorithm 1 tACT Construction

1: $\mathsf{tACT.Setup}(1^\lambda, t, N)$
2: Run $\mathsf{pp}' \leftarrow \mathsf{tSW.Setup}(1^\lambda)$ and let $\mathsf{H}' : \{0,1\}^* \to \{C \subset [1, n] \mid |C| = t' - 1\}$, where (t', n) are parameters of cut-and-choose. All hashes are treated as random oracle. Let $\mathbb{I} = \{I_1, \ldots, I_N\}$ denote the set of authorized issuers.
3: The issuers run $(\mathsf{pk}, \{\mathsf{sk}_j\}_{j \in [1,N]}) \leftarrow \mathsf{tSW.KeyGen}(1^\lambda, t, N)$, where $I_j \in \mathbb{I}$ receives the share sk_j of tSW secret signing key sk and public key is set to $\mathsf{pk} = g^{\mathsf{sk}}$.
4: **Output:** $\mathsf{pp} \leftarrow (\mathsf{pp}', \mathsf{H}', \mathsf{pk})$, $\mathsf{prv}_j \leftarrow \mathsf{sk}_j$

5: $\underline{\mathsf{tACT.Register}(\mathsf{a})}$

6: Generate a commitment on a unique attribute a defined by the system at index $i^* = 0$ of all admissible attributes. Set $\mathsf{cm} = \mathsf{Ped.Commit}(\mathsf{a}; r)$, where $r \leftarrow\!\!\$\ \mathbb{F}_p$.

7: **Output:** $\mathsf{st_{Rg}} = \{\mathsf{a}, r\}, \mathsf{cm}$

8: $\underline{\mathsf{tACT.TokenRequest}(\mathsf{st_{Rg}}, \mathsf{cm})}$

9: For all $k \in [t'-1]$ uniformly sample $(\mathsf{a}_k, r_k) \in \mathbb{F}_p$ then set $\mathsf{cm}_k = \mathsf{Ped.Commit}(\mathsf{a}_k; r_k)$ and $R_k = \mathsf{pk}^{r_k}$.

10: For all $k \in [t', n]$, compute $\mathsf{cm}_k = \left(\mathsf{cm} \prod_{j \in [t'-1]} \mathsf{cm}_j{}^{-\ell_j^{[t'-1],k}(0)}\right)^{\ell_{t'}^{[t'-1],k}(0)^{-1}}$,

$R_k = \left(\mathsf{pk}^r \prod_{j \in [t'-1]} R_j^{-\ell_j^{[t'-1],k}(0)}\right)^{\ell_{t'}^{[t'-1],k}(0)^{-1}}$ where $\ell_j^{[t'-1],k}$ is the j-th Lagrange basis polynomial for the points $1, \ldots, t'-1, k$.

11: User samples a key $\mathsf{k} \leftarrow\!\!\$\ \mathbb{F}_p$ and outputs $\mathsf{cm_k} = \mathsf{Ped.Commit}(\mathsf{k}; r_\mathsf{k})$, where $r_\mathsf{k} \leftarrow\!\!\$\ \mathbb{F}_p$

12: **Output:** $\mathsf{blindRequest} \leftarrow \{\mathsf{cm}, \{\mathsf{cm}_k\}_{k\in[1,n]}, \mathsf{cm_k}\}$, $\mathsf{rand_a} \leftarrow \{\mathsf{st_{Rg}}, \{R_k\}_{k\in[1,n]}, \mathsf{k}, r_\mathsf{k}\}$

13: $\underline{\mathsf{tACT.tIssue}(\mathsf{blindRequest}, \mathsf{prv}_j)}$

14: Each $I_j \in \mathbb{I}$:

a: Parse $\mathsf{blindRequest} = \{\mathsf{cm}, \{\mathsf{cm}_k\}_{k\in[1,n]}, \mathsf{cm_k}\}$

b: Check if $\mathsf{cm} = \prod_{k\in[n]} \mathsf{cm}_k{}^{\ell_k^{[n]}(0)}$ where $\ell_j^{[n]}$ is the j-th Lagrange basis polynomial for points $[n]$

c: Compute σ_{jk} as $\sigma_{jk} = \mathsf{tSW_{Ped}.Sign}(\mathsf{sk}_j, \mathsf{cm}_k, i^*)$ and set $\mathsf{blindToken}_{jk} \leftarrow \sigma_{jk}$

15: **Output:** $\{\mathsf{blindToken}_{jk}\}_{k\in[1,n]}$

16: $\underline{\mathsf{tACT.(Aggregate, Unblind)}(\{\mathsf{blindToken}_{jk}\}_{j\in[t],k\in[1,n]}, \mathsf{rand_a})}$

17: Collect a threshold t number of partial $\mathsf{blindToken}_{jk}$ on each a_k

18: Combine partial signatures: $\sigma_k = \mathsf{tSW.Comb}(\{\sigma_{jk}\}_{j\in[t]})$

19: Unblind σ_k by computing $s_k = \sigma_k \cdot R_k^{-1}$

20: Compute $s = \prod_{k\in[t']} s_k^{\ell_k^{[t']}(0)}$ where $\ell_j^{[t']}$ denotes the j-th Lagrange basis polynomial for the points $[t']$.

21: **Output:** $\mathsf{token} \leftarrow s, \{s_k\}_{k\in[1,n]}$

22: $\underline{\mathsf{tACT.Prove}(\mathsf{token}, \{s_k\}_{k\in[1,n]}, \mathsf{rand_a})}$

23: Compute $C \leftarrow \mathsf{H}'(\mathsf{token})$

24: Set $\{s'_k = s_k^\mathsf{k}\}_{k\in[1,n]}$, $\{R'_k = R_k^\mathsf{k}\}_{k\in C}$, and $\mathsf{pk}' = \mathsf{pk}^\mathsf{k}$

25: Generate $\pi = \mathsf{ZKPoK}\{(\mathsf{a}, \mathsf{k}, r, r_\mathsf{k}) : \mathsf{cm} = \mathsf{Ped.Commit}(\mathsf{a}; r) \wedge \mathsf{cm_k} = \mathsf{Ped.Commit}(\mathsf{k}; r_\mathsf{k}) \wedge \mathsf{pk}' = \mathsf{pk}^\mathsf{k}\}$

26: **Output:** $\mathsf{token}, \pi_\mathsf{token} \leftarrow \{\{s'_k\}_{k\in[1,n]}, C, \{R'_k\}_{k\in C}, \mathsf{pk}', \pi\}$

27: $\underline{\mathsf{tACT.Verify}(\mathsf{token}, \pi_\mathsf{token}, \mathsf{blindRequest})}$

28: Parse $\pi_\mathsf{token} = \{\{s'_k\}_{k\in[1,n]}, C, \{R'_k\}_{k\in C}, \mathsf{pk}', \pi\}$

29: If any of the following conditions is met, set $\mathsf{bit} \leftarrow 0$. Otherwise, set $\mathsf{bit} \leftarrow 1$.

a: $e(\mathsf{token}, \mathsf{pk}') \neq e(\prod_{k\in[t']} s'^{\ell_k^{[t']}(0)}, \mathsf{pk})$ where $\ell_j^{[t']}$ is the j-th Lagrange basis polynomial for points $[t']$ {to ensure the token's well-formedness.}

b: $\mathsf{ZKVerify}((\mathsf{cm}, \mathsf{cm_k}, \mathsf{pk}'), \pi) = 0$

c: $C \neq \mathsf{H}'(\mathsf{token})$ {cut-and-choose verification.}

d: $\exists\ k' \notin C : \prod_{c_j \in C} \mathsf{cm}_{c_j}{}^{\ell_j^{C,k'}(0)} \cdot \mathsf{cm}_{k'}{}^{\ell_{t'}^{C,k'}(0)} \neq \mathsf{cm}$ where $\ell_j^{C,k'}$ denotes the j-th Lagrange basis polynomial for the points $c_1, \ldots, c_{t'-1}, k'$ where $C = \{c_1, \ldots, c_{t'-1}\}$.

e: $\exists\ k \in C : \mathsf{tSW_{Ped}.Verify}(\mathsf{pk}', \mathsf{cm}_k, i^*, R'_k \cdot s'_k) = 0$

30: **Output:** bit

4 Building a DID System

4.1 Definitions and Properties

Subsequently, we formalize a Sybil-Resistance Self-Sovereign Identity (S3ID) system and provide a precise definition of its essential security properties.

Definition 5 (S3ID). *A Sybil-resistant self-sovereign identity scheme comprises a set of algorithms that are defined as follows (public parameters* pp *are assumed to be an implicit input to all algorithms):*

- $(\mathsf{pp}, \{\mathsf{prv}_j\}_{j\in[1,N]}) \leftarrow \mathsf{Setup}(1^\lambda, t, N)$*: On input security parameter* 1^λ*, and* t, N *representing the threshold and overall number of issuers it outputs system public parameters* pp *and private parameters*[10] prv_j *for issuer* $I_j, j \in [1, N]$*.*
- $(\mathsf{pp}_\mathsf{U}, \mathsf{prv}_\mathsf{U}, \mathsf{T}_\mathsf{Dedup}) \leftarrow \mathsf{Dedup}(\mathsf{a}, \{\mathsf{prv}_j\}_{j\in[1,N]}, \mathsf{T}_\mathsf{Dedup})$*: Is an interactive protocol between a user and the issuers. User input is private unique attribute* a*. Issuers inputs are private parameters* $\{\mathsf{prv}_j\}_{j\in[1,N]}$*. This algorithm generates user public and private parameters respectively denoted as* pp_U *and* prv_U*. In addition, the system updates the deduplication table* $\mathsf{T}_\mathsf{Dedup}$*, which serves as a database of redeemed tokens. This table helps to ensure that each token can only be redeemed once.*
- $(\{\mathsf{s}_i\}_{i=1}^L, \mathsf{T}_\mathsf{Dedup}) \leftarrow \mathsf{MicroCred}(\{\mathsf{a}_i\}_{i=1}^L, \mathsf{pp}_\mathsf{U}, \mathsf{prv}_\mathsf{U}, \{\mathsf{prv}_j$ $\}_{j\in[1,N]}, \mathsf{T}_\mathsf{Dedup})$*: Is an interactive protocol between a user and the issuers. User inputs are private attributes* $\{\mathsf{a}_i\}_{i=1}^L$*, and private parameters* prv_U*. Issuers inputs are private parameters* $\{\mathsf{prv}_j\}_{j\in[1,N]}$ *and user public parameters* pp_U*. The user obtains a set of micro-credentials* $\{\mathsf{s}_i\}_{i=1}^L$ *each embedding one of the private attributes* $\{\mathsf{a}_i\}_{i=1}^L$*.*
- $(\mathsf{cred}, \mathsf{tg}) \leftarrow \mathsf{AppCred}(\{\mathsf{a}_i\}_{i=1}^L, \{\mathsf{s}_i\}_{i=1}^L, \mathsf{prv}_\mathsf{U}, l, \phi)$*: This algorithm is run by the user who inputs private attributes* $\{\mathsf{a}_i\}_{i=1}^L$*, micro-credentials* $\{\mathsf{s}_i\}_{i=1}^L$*, private parameters* prv_U*, unique application message* l*, and application-specific statement* ϕ*. It outputs a concise credential* cred *certifying the satisfaction of* ϕ *by a subset of private attributes* $\{\mathsf{a}_i\}_{i=1}^L$*. Additionally, the user creates a tag* tg *which is unique per user per application and serves the purpose of enforcing the Sybil-resistance property.*
- $(\{0,1\}, \mathsf{T}_\mathsf{App}) \leftarrow \mathsf{VerifyCred}(\mathsf{cred}, \phi, \mathsf{tg}, \mathsf{T}_\mathsf{App})$*: This algorithm is run by any entity who wishes to verify if a credential embedding private attributes satisfies the application-specific statement* ϕ *and that the user showing it is the rightful owner. On input a credential* cred *and tag* tg*, it outputs bit* $b = 1$ *if it accepts the credential as valid and* $b = 0$ *otherwise. Also, it records* tg *in its database of redeemed tags* T_App*.*

Properties. An S3ID scheme requires to provide unforgeability, Sybil-resistance, and strong unlinkability. Similar to the tACT security definitions, unforgeability and Sybil-resistance are proven in the presence of an adversary $\mathcal{A}$ who can corrupt up to $t-1$ out of N issuers. Additionally, this adversary holds the ability

[10] This is usually run via a distributed key generation protocol.

to corrupt any number of users and applications within the system. For strong unlinkability, $\mathcal{A}$ presents the strongest category of adversaries who can corrupt *any* number of issuers and verifiers.

Unforgeability. This property captures the infeasibility of an adversarial user to generate credentials to prove a statement ϕ, where ϕ cannot be satisfied by any subset of attributes for which the adversary has obtained sufficiently many valid partial micro-credentials (including corrupted issuers). In interactive algorithms such as Dedup and MicroCred, we use subscript U and I to distinguish interactions on the user side and the issuer side, respectively.

Definition 6 (Unforgeability). *A S3ID scheme is unforgeable if for any PPT adversary $\mathcal{A}$ and any $T \geq 0$*

$$\mathbf{Adv}^{UF}_{S3ID,\mathcal{A}}(\lambda) := \Pr[UF_{S3ID,\mathcal{A}}(\lambda) \to 1] \leq \mathsf{negl}(\lambda),$$

where $UF_{S3ID,\mathcal{A}}(\lambda)$ is defined in Figure 6.

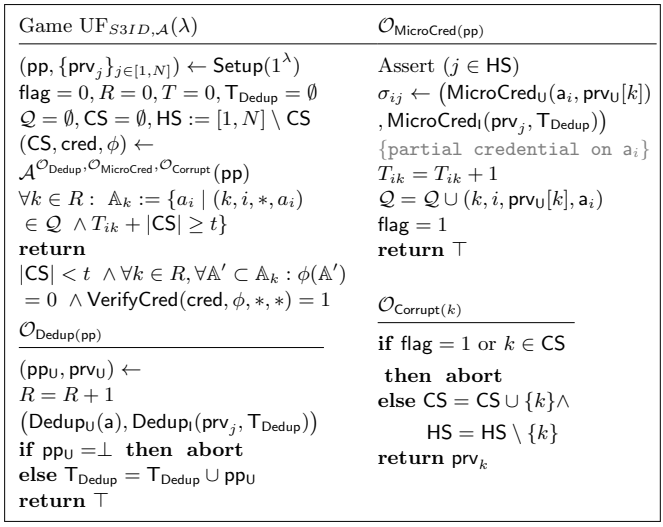

Fig. 6: Unforgeability experiment for S3ID.

Sybil-Resistance. This definition addresses the impossibility of an adversary to obtain more tokens than the number of users it controls during the deduplication mechanism, and therefore breaking the uniqueness of issued user identities. We refer to Figure 7 for the detailed game setup, where the adversary begins with initializing R users, each possessing a unique deduplication attribute. Subsequently, the adversary has the ability to request tokens for the users it controls. The adversary succeeds by outputting a database $\mathsf{T}_{\mathsf{Dedup}}$ including more than R valid tokens.

Definition 7 (Sybil-resistance). *A S3ID system is Sybil-resistant if for any PPT adversary $\mathcal{A}$ and any $R \geq 0$*

$$\mathbf{Adv}^{SR}_{S3ID,\mathcal{A},R}(\lambda) := \Pr[SR_{S3ID,\mathcal{A},R}(\lambda) \to 1] \leq \mathsf{negl}(\lambda),$$

where $SR_{S3ID,\mathcal{A},R}(\lambda)$ is defined in Figure 7.

Game $\mathrm{SR}_{S3ID,\mathcal{A},R}(\lambda)$	$\mathcal{O}_{\mathsf{Dedup}(\mathsf{pp})}$
$(\mathsf{pp}, \{\mathsf{prv}_j\}_{j\in[1,N]}) \leftarrow \mathsf{Setup}(1^\lambda)$ $R = 0, \mathsf{CS} = \emptyset$ $\mathsf{T}_{\mathsf{Dedup}} = \emptyset, \lvert\mathsf{T}_{\mathsf{Dedup}}\rvert = S$ $\mathsf{CS}, \mathsf{T}_{\mathsf{Dedup}}, \{\pi_i, \mathsf{bR}_i\}_{i\in[S]}$ $\leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dedup}}, \mathcal{O}_{\mathsf{Corrupt}}}(\mathsf{pp})$ **return** $\lvert\mathsf{CS}\rvert < t \wedge$ $S > R \wedge$ $(\forall\, i \in [S] \wedge \mathsf{token}_i \in \mathsf{T}_{\mathsf{Dedup}},$ $\mathsf{tACT.Verify}(\mathsf{token}_i, \pi_i, \mathsf{bR}_i) = 1)$	$R = R + 1$ $(\mathsf{pp}_\mathsf{U}, \mathsf{prv}_\mathsf{U}) \leftarrow (\mathsf{Dedup}_\mathsf{U}(\mathsf{a}),$ $\mathsf{Dedup}_\mathsf{I}(\mathsf{prv}_j, \mathsf{T}_{\mathsf{Dedup}}))$ **if** $\mathsf{pp}_\mathsf{U} = \perp$ **then abort** **else** $\mathsf{T}_{\mathsf{Dedup}} = \mathsf{T}_{\mathsf{Dedup}} \cup \mathsf{pp}_\mathsf{U}$ **return** $\mathsf{T}_{\mathsf{Dedup}}$ $\mathcal{O}_{\mathsf{Corrupt}(k)}$ **if** $k \in \mathsf{CS}$ **then abort** **else** $\mathsf{CS} = \mathsf{CS} \cup \{k\} \wedge$ $\mathsf{HS} = \mathsf{HS} \setminus \{k\}$ **return** prv_k

Fig. 7: Sybil resistance experiment for S3ID.

Strong Unlinkability. It should be infeasible for an adversarial verifier potentially colluding with any number of issuers and verifiers to link the execution of AppCred with either another execution of AppCred (unlinkability across applications), or with the executions of MicroCred and Dedup algorithms for a specific set of attributes. The $\mathrm{SUNLINK}_{S3ID,\mathcal{A}}(\lambda)$ game is defined in Figure 8. The adversary $\mathcal{A}$ can instruct users in the system with a specific set of attributes to generate tokens and micro-credentials $\{\mathsf{s}_i\}_{i=1}^{L}$ respectively through the $\mathcal{O}_{\mathsf{Dedup}}$ and $\mathcal{O}_{\mathsf{MicroCred}}$ oracles. Then $\mathcal{A}$ chooses two distinct sets $(\mathsf{token}_\mathsf{b}, \{\mathsf{s}_{i,b}\}_{i=1}^{L}), b \in \{0,1\}$, among which one of them is randomly picked to generate cred, through the $\mathcal{O}_{\mathsf{AppCred}}$ oracle. The adversary should distinguish cred for two different client/attribute pairs; or it needs to distinguish cred for the same client but two different AppCred sessions. The definition requires that VerifyCred succeeds on both the credentials generated based on requests provided by the adversary.

Definition 8 (Strong Unlinkability). *A S3ID system achieves Strong Unlinkability if for PPT adversary $\mathcal{A}$ the advantage defined as*

$$\mathbf{Adv}^{SUNLINK}_{S3ID,\mathcal{A}}(\lambda) := |2\Pr[SUNLINK_{S3ID,\mathcal{A}}(\lambda) \to 1] - 1|,$$

where $SUNLINK_{S3ID,\mathcal{A}}(\lambda)$ is defined in Figure 8, is negligible, i.e.,

$$\mathbf{Adv}^{SUNLINK}_{S3ID,\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda).$$

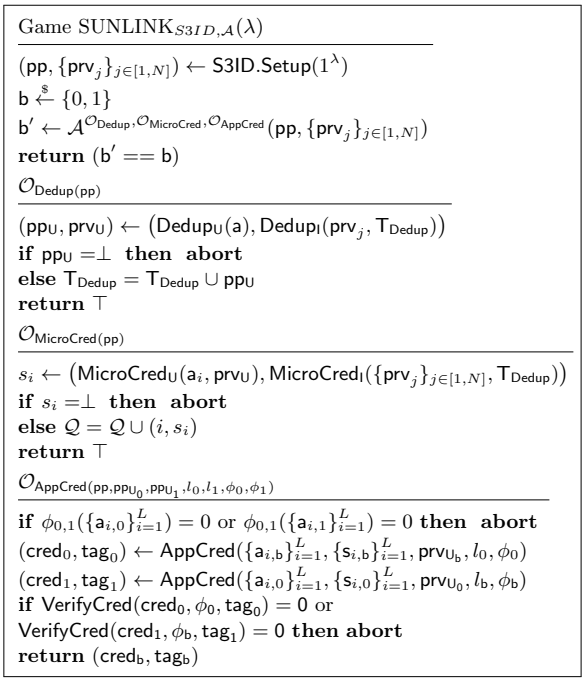

Fig. 8: Strong unlinkability experiment for S3ID.

Besides the formal security properties, we also want the following properties to hold:

Non-Transferability. To prevent the transfer of credentials among users, it needs to be ensured that issued credentials can only be released by the legitimate owner and cannot be transferred to any other individual.

Non-Interactivness. After establishing consensus on system parameters in the key distribution and setup phase, the issuers can function independently without the need to further synchronize or coordinate. Additionally, the system enables users to generate unique application-specific credentials without the necessity of interacting with the issuers for each transaction.

4.2 S3ID Design

Now we present an application of our tACT system by employing it as a fundamental building block for our S3ID scheme. This is detailed in Algorithm 2. In the S3ID.Dedup algorithm, a token token generated in the underlying tACT algorithm serves as a distinctive high-entropy function derived from the unique attribute a, enabling user deduplication. Upon successful verification of token, the issuers

proceed by comparing token against their database of redeemed tokens and reject if the value already occurs there. Otherwise, they update the database by adding the redeemed token to $\mathsf{T}_{\mathsf{Dedup}}$. In $\mathsf{T}_{\mathsf{Dedup}}$, alongside the redeemed token, issuers also keep a commitment cm_{k} to the unique key k registered by user, and a binary vector $\mathbb{IDX}$ of length L initially set to all zero, where $\mathbb{IDX}[i] = 1$ shows the user is issued a micro-credential for her attribute a_i (see $\mathsf{S3ID.MicroCred}$ for more details).

In the $\mathsf{S3ID.MicroCred}$ algorithm, the user, to acquire anonymous micro-credentials for its desired attribute a_i, generates a commitment cm_i on $(\mathsf{k}, \mathsf{a}_i)$, and a proof of knowledge π_i to prove the consistency of cm_i with the commitment cm_{k} on the registered key. Then she submits $\{\mathsf{token}, \mathsf{cm}_i, \pi_i\}$ to the issuers. Issuer I_j checks the latest version of deduplication table $\mathsf{T}_{\mathsf{Dedup}}$. If $\mathsf{token} \in \mathsf{T}_{\mathsf{Dedup}}$ and $\mathbb{IDX}[i] = 0$ and the proof π_i passes the verification, it generates a partial $\mathsf{tSW}_{\mathsf{Ped}}$ signature on commitment cm_i to a_i bound to user-unique key k. The user then proceeds to gather sufficient partial signatures, combines them to form a full signature, and converts the result to a tSW signature subsequently. The resulting s_i is a micro-credential on attribute a_i which is bound to key k.

In the $\mathsf{S3ID.AppCred}$ algorithm, the user is empowered to generate application-specific credentials. We assume a unique message l for each application that specifies the application name (e.g., "participate in auction X"). Also consider $\mathbb{A}_l = \{\mathsf{a}_i\}_{i \in \mathbb{Q}}, \mathbb{Q} \subset [1, L]$ as the subset of attributes corresponding to the application ($\mathbb{A}_l \subset \mathbb{A}$) and ϕ as the application-specific statement (or predicate). To get an application-specific credential for an application with message l, the user locally aggregates the relevant micro-credentials encapsulating attributes in $\mathbb{A}_l$ with respect to the message l, and generates one small application credential ζ, which are the aggregated signatures masked by pk^r for uniformly random r. The user also generates τ as a Pedersen commitment to the designated attributes together with the registered key k using randomness r. The presence of k in τ enforces users to build application credentials based on their genuinely owned micro-credentials, reinforcing non-transferability.

Algorithm 2 S3ID Construction

1: $\mathsf{S3ID.Setup}(1^\lambda, t, N)$
2: Let $\mathbb{A} = \{\mathsf{a}, \mathsf{a}_1, \ldots, \mathsf{a}_L\}$ denote the set of admissible attributes and $\mathbb{I} = \{I_1, \ldots, I_N\}$ denote the set of authorized issuers.{unique attribute a is defined at index 0.}
3: Run $(\mathsf{pp}', \{\mathsf{prv}_j\}_{j=1}^N) \leftarrow \mathsf{tACT.Setup}(1^\lambda, t, N)$ and $(\mathsf{pp}'' = \{g, u_0, u_1, \ldots, u_L\}) \leftarrow \mathsf{Ped.Setup}(1^\lambda, L+1)$. Let $\mathsf{H}'' : \{0,1\}^* \to \mathbb{F}_p$ and $\mathsf{h} : \mathbb{G}_1 \to \mathbb{F}_p$ be secure hash functions.
4: Issuers initialize a shared table $\mathsf{T}_{\mathsf{Dedup}} := \emptyset$
5: **Output:** System public parameters $\mathsf{pp} = (\mathsf{pp}', \mathsf{pp}'', \mathsf{H}'', \mathsf{h}, \mathsf{T}_{\mathsf{Dedup}})$, issuers' private parameter $\{\mathsf{prv}_j\}_{j \in [1,N]}$

6: $\mathsf{S3ID.Dedup}(\mathsf{a}, \{\mathsf{prv}_j\}_{j\in[1,N]}, \mathsf{T}_{\mathsf{Dedup}})$

7: The user on input private unique attribute a and issuers on input $\{\mathsf{prv}_j\}_{j\in[1,N]}$ interact in the following way:
a: User runs $(\mathsf{st}_{\mathsf{Rg}}, \mathsf{cm}) \leftarrow \mathsf{tACT.Register}(\mathsf{a}), (\mathsf{bR}, \mathsf{rand}_{\mathsf{a}}) \leftarrow \mathsf{tACT.TokenRequest}(\mathsf{st}_{\mathsf{Rg}}, \mathsf{cm})$
b: I_j runs $\{\mathsf{bT}_{jk}\}_{k\in[1,n]} \leftarrow \mathsf{tACT.tIssue}(\mathsf{bR}, \mathsf{prv}_j)$
c: User runs $\mathsf{token}, \{s_k\}_{k\in[1,n]} \leftarrow \mathsf{tACT.(Aggregate, Unblind)}(\{\mathsf{bT}_{jk}\}_{j\in[t]}, \mathsf{rand}_{\mathsf{a}})$
d: User runs $\pi \leftarrow \mathsf{tACT.Prove}(\mathsf{token}, \{s_k\}_{k\in[1,n]}, \mathsf{rand}_{\mathsf{a}})$
e: User samples a key $\mathsf{k} \leftarrow\!\!\$\ \mathbb{F}_p$ and outputs $\mathsf{cm}_{\mathsf{k}} = \mathsf{Ped.Commit}(\mathsf{k}; r_{\mathsf{k}})$, where $r_{\mathsf{k}} \leftarrow\!\!\$\ \mathbb{F}_p$
f: Issuers run $\mathsf{bit} \leftarrow \mathsf{tACT.Verify}(\mathsf{token}, \pi, \mathsf{bR})$
g: If $(\mathsf{bit} = 0 \vee (\mathsf{bit} = 1 \wedge \mathsf{token} \in \mathsf{T}_{\mathsf{Dedup}}))$, abort.
h: Initialize binary vector $\mathbb{IDX}$ of length L with all zeros.
i: Set $\mathsf{T}_{\mathsf{Dedup}} \leftarrow \mathsf{T}_{\mathsf{Dedup}} \cup \{\mathsf{token}, \mathsf{cm}_{\mathsf{k}}, \mathbb{IDX}\}$.

8: **Output:** $\mathsf{pp}_{\mathsf{U}} \leftarrow \{\mathsf{token}, \mathsf{cm}_{\mathsf{k}}, \mathbb{IDX}\}, \mathsf{prv}_{\mathsf{U}} \leftarrow \{\mathsf{a}, \mathsf{k}\}, \mathsf{T}_{\mathsf{Dedup}}$

9: $\mathsf{S3ID.MicroCred}(\{\mathsf{a}_i\}_{i=1}^{L}, \mathsf{pp}_{\mathsf{U}}, \mathsf{prv}_{\mathsf{U}}, \{\mathsf{prv}_j\}_{j\in[1,N]}, \mathsf{T}_{\mathsf{Dedup}})$

10: The user on input $\mathsf{prv}_{\mathsf{U}}$ and private attribute a_i for which she seeks to obtain a micro-credential, and issuers on input $\{\mathsf{prv}_j\}_{j\in[1,N]}$ interact in the following way:
a: User picks random $r_i \leftarrow\!\!\$\ \mathbb{F}_p$ and computes $\mathsf{cm}_i = \mathsf{Ped.Commit}(\mathsf{k}, \mathsf{a}_i; r_i) = g^{r_i} \cdot u_0^{\mathsf{k}} \cdot u_i^{\mathsf{a}_i}$.
b: Generates $\pi_i = \mathsf{ZKPoK}\{(\mathsf{a}_i, \mathsf{k}, r_i, r_{\mathsf{k}}) : \mathsf{cm}_i = \mathsf{Ped.Commit}(\mathsf{k}, \mathsf{a}_i; r_i) \wedge \mathsf{cm}_{\mathsf{k}} = \mathsf{Ped.Commit}(\mathsf{k}; r_{\mathsf{k}}) \wedge \phi(\mathsf{a}_i) = 1\}$
c: Outputs $\mathsf{request}_i \leftarrow \{\mathsf{token}, \mathsf{cm}_i, \pi_i\}$
d: I_j parses $\mathsf{request}_i = \{\mathsf{token}, \mathsf{cm}_i, \pi_i\}$
e: If $(\mathsf{token} \in \mathsf{T}_{\mathsf{Dedup}} \wedge \mathbb{IDX}[i] = 0)$, go to next step. Else, abort.
f: If $\mathsf{ZKVerify}((\mathsf{cm}_i, \mathsf{cm}_{\mathsf{k}}, \phi), \pi_i) = 0$, abort.
g: Signs $\sigma_{ij} = \mathsf{tSW}_{\mathsf{Ped}}.\mathsf{Sign}(\mathsf{prv}_j, \mathsf{cm}_i, i)$ and sets $\mathbb{IDX}[i] \leftarrow 1$
h: User computes full signature $\sigma_i = \mathsf{tSW.Comb}(\{\sigma_{ij}\}_{j\in[t]})$
i: Unblinds σ_i by computing $s_i = \sigma_i \cdot \mathsf{pk}^{-r_i}$

11: **Output:** $s_i, \mathsf{T}_{\mathsf{Dedup}}$

12: $\mathsf{S3ID.AppCred}(\{\mathsf{a}_i\}_{i=1}^{L}, \{\mathsf{s}_i\}_{i=1}^{L}, \mathsf{prv}_{\mathsf{U}}, l, \phi, \mathbb{A}_l)$

13: The application relying party initializes a table $\mathsf{T}_{\mathsf{App}} := \emptyset$

14: The user, on input unique application message l, application-specific statement ϕ, and $\mathbb{A}_l = \{\mathsf{a}_i\}_{i\in\mathbb{Q}}, \mathbb{Q} \subset [1, L]$ as the subset of attributes corresponding to the application, follows these steps:
a: Picks $r \leftarrow\!\!\$\ \mathbb{F}_p$ and calculates $\tau = g^r \cdot u_0^{|\mathbb{Q}|\mathsf{k}} \cdot \prod_{i\in\mathbb{Q}} u_i^{\mathsf{a}_i}$
b: Generates $\zeta = \mathsf{pk}^r \cdot \prod_{i\in\mathbb{Q}} s_i$
c: Sets $\mathsf{tg} = \mathsf{PRF}(\mathsf{k}, l)$
d: Generates $\pi = \mathsf{ZKPoK}\{(\mathbb{A}_l, \mathsf{k}, \zeta, r) : \tau = g^r \cdot u_0^{|\mathbb{Q}|\mathsf{k}} \cdot \prod_{i\in\mathbb{Q}} u_i^{\mathsf{a}_i} \wedge \mathsf{tg} = \mathsf{PRF}(\mathsf{k}, l) \wedge \phi(\mathbb{A}_l) = 1 \wedge e(\zeta, g) = e(\prod_{i\in\mathbb{Q}} \mathsf{H}(i) \cdot \tau, \mathsf{pk})\}$

15: **Output:** $\mathsf{cred} \leftarrow \{\tau, \pi\}, \mathsf{tg}$

16: $\mathsf{S3ID.VerifyCred}(\mathsf{cred}, l, \phi, \mathsf{tg}, \mathsf{T}_{\mathsf{App}})$

17: The verifying party does the following:

18: Parses $\mathsf{cred} = \{\tau, \pi\}$

19: If all of the following conditions are met, set $\mathsf{bit} \leftarrow 1$. Otherwise, set $\mathsf{bit} \leftarrow 0$.
a: Tag $\mathsf{tg} \notin \mathsf{T}_{\mathsf{App}}$ {$\mathsf{T}_{\mathsf{App}}$ = database of redeemed tags.}
b: $\mathsf{ZKVerify}((\tau, \mathsf{tg}, l, \phi), \pi) = 1$

20: **Output:** $\mathsf{bit}, \mathsf{T}_{\mathsf{App}} \leftarrow \mathsf{T}_{\mathsf{App}} \cup \mathsf{tg}$

Furthermore, micro-credentials are bound to unique attribute indices used by the verifier in the verification equation. This safeguards against replace attacks and prevents users from acquiring application credentials by substituting a different combination of micro-credentials with encoded attributes that do not belong to $\mathbb{A}_l$. To prevent double submissions and achieve Sybil-resistance, the user generates a tag tg as a PRF evaluated under the registered key k on the unique application message l. Therefore, tg is unique per user and per application. The PRF should have algebraic properties such that they can be easily be proven using efficient ZKPoK, i.e., Schnorr proofs in our case. Two possible instantiations of such PRFs are the Naor-Pinkas-Reingold PRF [NPR99] $\mathsf{PRF}(\mathsf{k}, l) = \mathsf{H}(l)^{\mathsf{k}}$ or Dodis-Yampolskiy PRF [DY05] $\mathsf{PRF}(\mathsf{k}, l) = g^{(\mathsf{k}+l)^{-1}}$, where for our instantiation we choose the former. The user then computes a ZKPoK proof π to demonstrate the correct evaluation of tg on the message l with respect to the key k encapsulated in all of the user's micro-credentials. Additionally, this proof serves to confirm that the user claiming these attributes is indeed the rightful owner. Moreover, the user proves the validity of the masked tSW signatures ζ without revealing it, which amounts to the pairing equation $e(\zeta, g) = e(\prod_{i \in \mathbb{Q}} \mathsf{H}(i) \cdot \tau, \mathsf{pk})$ and can be efficiently proven using a Groth-Sahai ZKPoK proof [GS12] (see Section 5 for a more detailed discussion about the proven statement). Finally, the user sets $\mathsf{cred} = (\tau, \pi, \mathsf{tg})$ as its self-sovereign application-specific credential and sends it to the relying party.

In $\mathsf{S3ID.VerifyCred}$ algorithm, any party who knows public parameters pp can verify a user's credential cred. The verifier first checks if the tag tg does not occur in the unique per application database $\mathsf{T}_{\mathsf{App}}$ of submitted tags, and subsequently verifies the proof π. If the verification succeeds, the verifier is simultaneously convinced that all the micro-credentials contained within the application credential have been appropriately signed by a threshold of issuers.

Remark. We note that establishing a one-to-one correspondence between the user and the claimed attributes can be achieved through the utilization of privacy-preserving oracles [ZCC+16, ZMM+20]. In this context, our systems can seamlessly integrate with a legacy-compatible oracle system. This integration enables users to provide an additional oracle proof, denoted as $\pi^{\mathcal{O}}$, thus substantiating ownership of the attribute a concealed within the commitment cm, ultimately demonstrating unique personhood while maintaining privacy.

For the security analysis of $\mathsf{S3ID}$, we refer to Appendix B.2.

5 Evaluation and Experimental Results

We implemented tACT from Section 3 and $\mathsf{S3ID}$[11] from Section 4.2 in Rust using the `ark-bls12-381`[12] and `groth-sahai-rs`[13] crate for implementations of the

[11] See https://anonymous.4open.science/r/s3id-4DC0/ for the code.
[12] https://crates.io/crates/ark_bls12_381, version 0.3.0.
[13] https://github.com/jdwhite48/groth-sahai-rs

Table 2: tACT runtime in milliseconds with $t' = \frac{n}{2} + 1$.

$N = 4, t = \frac{N}{2} + 1$	$n = 30$	$n = 40$	$n = 128$
Register	1.32	1.52	1.58
TokenRequest	387.0	583.4	5750
tIssue	37.8	50.12	167.0
(Aggregate, Unblind)	23.3	31.10	103.0
Prove	42.2	54.57	173.4
Verify	251.6	412.8	3552
$N = 64, t = \frac{N}{2} + 1$	$n = 30$	$n = 40$	$n = 128$
Register	1.57	1.56	1.57
TokenRequest	393.9	578.9	5686
tIssue	38.5	51.5	164.3
(Aggregate, Unblind)	153.5	203.1	622.7
Prove	42.8	56.4	159.7
Verify	253.9	407.8	3495

pairing-friendly BLS12-381 curve and the Groth-Sahai proof system. As this curve provides a Type-3 pairing, we applied a generic compiler [AGH15, AHO16] at the cost of duplicating operations and values in the second source group. With the implementation, we evaluated the runtime costs of the individual algorithms. As discussed in Section 3.2, the parameters were set to $t = \frac{N}{2} + 1$ and $t' = \frac{n}{2} + 1$. For the number of issuers, N, we considered values of 4 (as in the evaluation of CanDID) and 64 and for n we compared choices of 30, 40, and 128.

The benchmarks were performed with an Intel Core i7-1265U and 16 GB of RAM. The runtime results of tACT are depicted in Table 2. We observe that the biggest impact on the performance is the choice of t' and n. This observation matches the intuition based on the construction where the TokenRequest, tIssue, (Aggegate, Unblind), Prove and Verify linearly depend on t' or n. Notably, only (Aggegate, Unblind) also depend linearly on t whereas tIssue is executed in parallel at the N issuers.

For S3ID, the benchmark results are presented in Table 3. We observe that for algorithms Dedup and MicroCred the runtime reflects the fact that both depend on the number of issuers and the security parameter. Notably, for a larger security parameter and more issuers, the runtime of Dedup is ≈ 10 seconds at the same magnitude as CanDID [MMZ+21] while the latter uses a significantly smaller security parameter and only 4 issuers. Also note that both AppCred and VerifyCred are independent of the choices of the parameters and the variations are due to noise on the benchmarking system. The implementation currently does not include the evaluation of ϕ as the policies depend on the application domain. Concrete policies and the representation of ϕ will have an non-negligible impact on the performance. Hence, the runtimes AppCred and VerifyCred are lower bounds.

Table 3: S3ID runtime in milliseconds with $t' = \frac{n}{2} + 1$.

$N = 4, t = \frac{N}{2} + 1$	$n = 30$	$n = 40$	$n = 128$
Dedup	782.6	1186	9763
MicroCred	165.2	166.3	166.3
AppCred	34.1	33.6	34.1
VerifyCred	53.6	32.4	43.5
$N = 64, t = \frac{N}{2} + 1$	$n = 30$	$n = 40$	$n = 128$
Dedup	1554	2217	13179
MicroCred	2008	2002	2001
AppCred	34.3	34.2	35.3
VerifyCred	43.9	43.0	42.8

In the implementation of AppCred and VerifyCred we also split the NIZK proof into a Groth-Sahai proof of the pairing equation $e(\zeta, g) = e(\prod_{i \in \mathbb{Q}} \mathsf{H}(i) \cdot \tau, \mathsf{pk})$ and a Schnorr proof of the other statements – the well-formedness of $\tau = g^r u_0^{|\mathbb{Q}|\mathsf{k}} \prod_{i \in \mathbb{Q}} u_i^{\mathsf{a}_i}$ and the evaluation of the PRF $\mathsf{tg} = \mathsf{PRF}(\mathsf{k}, l) = \mathsf{H}(l)^{\mathsf{k}}$. To bind the two proofs together and to ensure the non-malleability of the Groth-Sahai proof, the Groth-Sahai proof is included in the hash of the Fiat-Shamir transform of the Schnorr proof.

Furthermore, while we observe an increase in the runtimes of Dedup and MicroCred as the number of issuers grows, the protocol can be scaled to accommodate more issuers without significantly impacting overall runtime. This is because these algorithms are parallelizable, single-round, and require only one-time execution per user. The increase in the benchmarks is caused by running all issuers on a system with a higher number of issuers than number of cores.

6 Conclusion

This paper first introduces tACT, marking a shift in anonymous counting tokens by operating in a distributed environment. Utilizing tACT as a core element, we present a system called S3ID that achieves distributed threshold issuance, unlinkable multi-show selective disclosure, non-interactive, and non-transferable credentials. We provide an implementation of our system along with benchmarks which show that our work addresses the shortcomings of current self-sovereign identity systems.

Acknowledgments. This work received funding from the EU research project SWARMCHESTRATE (project no. 101135012) and the European Union's Horizon Europe project SUNRISE (project no. 101073821), and by PREPARED, a project funded by the Austrian security research programme KIRAS of the Federal Ministry of Finance (BMF).

References

AGH15. Joseph A. Akinyele, Christina Garman, and Susan Hohenberger. Automating fast and secure translations from type-i to type-iii pairing schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1370–1381. ACM, 2015.

AGOT14. Masayuki Abe, Jens Groth, Miyako Ohkubo, and Takeya Tango. Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 241–260. Springer, 2014.

AHO16. Masayuki Abe, Fumitaka Hoshino, and Miyako Ohkubo. Design in type-i, run in type-iii: Fast and scalable bilinear-type conversion using integer programming. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 387–415. Springer, 2016.

BBG+20. James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

BCC+21. Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, 1:1–136, 2021.

BCK+22. Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 517–550. Springer, 2022.

BIK+17. Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

BKKJ+17. Maria Borge, Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, and Bryan Ford. Proof-of-personhood: Redemocratizing permissionless cryptocurrencies. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 23–26. IEEE, 2017.

BL22. Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 193–207. ACM, 2022.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology– ASIACRYPT 2001*, volume 2248, pages 514–532. Springer, 2001.

BRS23. Fabrice Benhamouda, Mariana Raykova, and Karn Seth. Anonymous counting tokens. In *ASIACRYPT*, 2023.

C+16. Christian Cachin et al. Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, pages 1–4. Chicago, IL, 2016.

CCK+23. Panagiotis Chatzigiannis, Konstantinos Chalkias, Aniket Kate, Easwar Vivek Mangipudi, Mohsen Minaei, and Mainack Mondal. Sok: Web3 recovery mechanisms. *Cryptology ePrint Archive*, 2023.

CDL+13. Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. Concepts and languages for privacy-preserving attribute-based authentication. In *Policies and Research in Identity Management: Third IFIP WG 11.6 Working Conference, IDMAN 2013, London, UK, April 8-9, 2013. Proceedings 3*, pages 34–52. Springer, 2013.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994.

CKM23. Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 678–709. Springer, 2023.

CKS24. Elizabeth Crites, Aggelos Kiayias, and Amirreza Sarencheh. Syra: Sybil-resilient anonymous signatures with applications to decentralized identity. *Cryptology ePrint Archive*, 2024.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology–CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings 24*, pages 56–72. Springer, 2004.

CLPK22. Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In *Public-Key Cryptography–PKC 2022: 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8–11, 2022, Proceedings, Part I*, pages 409–438. Springer, 2022.

Cra96. Ronald Cramer. Modular design of secure yet practical cryptographic protocols. *Ph. D.-thesis, CWI and U. of Amsterdam*, 2, 1996.

CS97. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich*, 260, 1997.

DGS+18. 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.

dif20. Decentralized Identity Foundation. https://identity.foundation/, 2020.

DKL+23. Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. Threshold BBS+ signatures for distributed anonymous credential issuance. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 773–789. IEEE, 2023.

DY05. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography-PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005. Proceedings 8*, pages 416–431. Springer, 2005.

fac18. Facebook data breach. https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html, 2018.

FG18. Georg Fuchsbauer and Romain Gay. Weakly secure equivalence-class signatures from standard assumptions. In *Public-Key Cryptography–PKC 2018: 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II 21*, pages 153–183. Springer, 2018.

FHS19. 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.

FS86. 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.

Gol09. Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

GS12. Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

HIJ+21. Sharon Huang, Subodh Iyengar, Sundar Jeyaraman, Shiv Kushwah, Chen-Kuei Lee, Zutian Luo, Payman Mohassel, Ananth Raghunathan, Shaahid Shaikh, Yen-Chieh Sung, et al. Dit: De-identified authenticated telemetry at scale, 2021.

HP23. Chloé Hébant and David Pointcheval. Traceable constant-size multi-authority credentials. *Information and Computation*, page 105060, 2023.

HS21. 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.

KHG12. Aniket Kate, Yizhou Huang, and Ian Goldberg. Distributed key generation in the wild. *Cryptology ePrint Archive*, 2012.

KLOR20. Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part I 40*, pages 308–336. Springer, 2020.

MBG+23. Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In *CCS*, pages 30–44. ACM, 2023.

MMZ+21. Deepak Maram, Harjasleen Malvai, Fan Zhang, Nerla Jean-Louis, Alexander Frolov, Tyler Kell, Tyrone Lobban, Christine Moy, Ari Juels, and Andrew Miller. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1348–1366. IEEE, 2021.

ms23. Microsoft data breach. https://www.reuters.com/world/us/chinese-hackers-stole-60000-emails-us-state-department-microsoft-hack-senate-2023-09-27/, 2023.

MSM23. Omid Mir, Daniel Slamanig, and René Mayrhofer. Threshold delegatable anonymous credentials with controlled and fine-grained delegation. *IEEE Transactions on Dependable and Secure Computing*, pages 1–16, 2023.

NPR99. Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99*, volume 1592, pages 327–346. Springer, 1999.

Ped91. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

PS16. 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.

SABB+18. 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.

San20. Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In *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*, pages 628–656. Springer, 2020.

SKP+23. Tanusree Sharma, Yujin Kwon, Kornrapat Pongmala, Henry Wang, Andrew Miller, Dawn Song, and Yang Wang. Unpacking how decentralized autonomous organizations (daos) work in practice. *arXiv preprint arXiv:2304.09822*, 2023.

Sov18. Sovrin. Public service utility enabling self-sovereign identity on the internet. https://sovrin.org/, 2018.

SW13. Hovav Shacham and Brent Waters. Compact proofs of retrievability. *Journal of cryptology*, 26(3):442–483, 2013.

TBM+20. Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. Verifiable timed signatures made practical. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1733–1750, 2020.

W3C18. W3C. Decentralized identifiers (DIDs). v0.11:data model and syntaxes for decentralized identifiers. https://w3c-ccg.github.io/did-spec/, 2018.

yah17. Yahoo data breach. https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html, 2017.

ZCC+16. Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pages 270–282, 2016.

ZMM+20. Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. Deco: Liberating web data using decentralized oracles for tls. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1919–1938, 2020.

A Additional Preliminaries

A.1 Notation

Let $\lambda \in \mathbb{N}$ represent the security parameter. For any finite set S, we use the notation $x \leftarrow S$ or $x \leftarrow\!\!\$\ S$ to indicate that x is uniformly sampled from S. For an algorithm A, we denote the process of running A on input x and utilizing uniformly random coins r as $y \leftarrow A(x;r)$, where the result is assigned to y. In this context, $[i,j]$ represents a set comprising all integers within the range from i to j, inclusive. We assume all algorithms are probabilistic polynomial-time (PPT) unless explicitly stated otherwise. Public parameters are assumed as an implicit input to all algorithms in a scheme. negl is a negligible function in the security parameter λ, that is for any positive integer k and for any large enough λ, $\mathsf{negl}(\lambda) \leq 1/\lambda^k$.

A.2 Pedersen Commitments

We adopt the Pedersen commitment as our commitment scheme with binding and hiding properties. We utilize Pedersen commitment over a group $\mathbb{G}$ of large order p in which the discrete logarithm assumption holds. The scheme is defined as a tuple of the following algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Ped.Setup}(1^\lambda)$: This algorithm inputs security parameter λ and outputs public commitment parameters pp as two generators g, u of the group $\mathbb{G}$.
- $\mathsf{cm} \leftarrow \mathsf{Ped.Commit}(m;r)$: This algorithm generates a commitment cm of message m using randomness r. The cm is of the form $\mathsf{cm} = g^r \cdot u^m$ where $r \leftarrow\!\!\$\ \mathbb{F}_p$.
- $\{0,1\} \leftarrow \mathsf{Ped.Open}(\mathsf{cm}, r, m)$: To open a commitment cm, the sender sends $\{r, m\}$ to the receiver. The receiver verifies the opening by checking whether $\mathsf{cm} = g^r \cdot u^m$ and outputs a bit $b \in \{0,1\}$.

The binding property of the commitment scheme necessitates that the prover lacks knowledge of the discrete log relation between generators g, u.

Generalized Pedersen Commitment. There is an important generalization of the Pedersen commitment scheme [Ped91], with which a prover can commit to multiple messages $(m_0, \ldots, m_{n-1})$ for $n \in \mathbb{Z}^+$. The generalized Pedersen commitment over a group G of prime order p is defined as a tuple of the following algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Ped.Setup}(1^\lambda, n)$: This algorithm inputs security parameter λ and outputs public commitment parameters pp consisting of $n+1$ elements of the group $\mathbb{G}$ denoted as $g, u_0, u_1, \ldots, u_{n-1}$.
- $\mathsf{cm} \leftarrow \mathsf{Ped.Commit}(m_0, \ldots, m_{n-1}; r)$: To commit to messages $(m_0, \ldots, m_{n-1})$, where $m_i \in \mathbb{F}_p$ for $i \in [0, n-1]$, the algorithm samples a random value $r \leftarrow\!\!\$\ \mathbb{F}_p$, computes $\mathsf{cm} = g^r \cdot \prod_{i=0}^{n-1} u_i^{\mathsf{m}_i}$ and outputs cm.
- $\{0,1\} \leftarrow \mathsf{Ped.Open}(\mathsf{cm}, r, m_0, \ldots, m_{n-1})$: The algorithm inputs opening of commitment cm as $\{r, m_0, \ldots, m_{n-1}\}$ and verifies the opening by checking whether $\mathsf{cm} = g^r \cdot \prod_{i=0}^{n-1} u_i^{\mathsf{m}_i}$. It outputs a bit $b \in \{0,1\}$.

A.3 Zero-Knowledge Proofs of Knowledge

We define zero-knowledge proofs of knowledge (ZKPoK) and discuss their non-interactive variants (NIZK). In this paper, we require protocols to prove knowledge of discrete logarithm relations. This can be efficiently realized by utilizing Sigma protocols [CDS94, Cra96], which are efficient instantiations of ZKPoK and can be made non-interactive using the Fiat-Shamir heuristic [FS86]. Moreover, we need to prove a pairing product equation and thus also rely on the Groth-Sahai NIZK proof system [GS12]. Let $\mathcal{R} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a NP-witness-relation with corresponding NP-language $\mathcal{L} := \{x : \exists w \; s.t. \; \mathcal{R}(x,w) = 1\}$. A non-interactive zero-knowledge proof (NIZK) system for $\mathcal{R}$ has the following algorithms:

- $\mathsf{crs} \leftarrow \mathsf{ZKSetup}(1^\lambda, \mathcal{R})$: Generates common reference string crs for the prove relationship $\mathcal{R}$.
- $\pi \leftarrow \mathsf{ZKPoK}(\mathcal{R}, \mathsf{crs}, x, w)$: Generates a proof π that the prover knows a witness w such that the input statement x satisfies the relation $\mathcal{R}(x,w)$.
- $\{0,1\} \leftarrow \mathsf{ZKVerify}(\mathsf{crs}, x, \pi)$: Efficiently verifies the correctness of the proof for a statement x. Outputs a bit $b \in \{0,1\}$, where $b = 1$ shows that proof π is accepted, and $b = 0$ otherwise.

The notation used to represent ZK proofs is $\mathsf{ZKPoK}\{w : \text{statement about } w\}$, as introduced by Camenisch *et al.* [CS97]. This notation indicates showing in zero-knowledge that certain secret values w fulfill the statement listed after the colon. A ZKPoK system must satisfy the following properties: (I) Zero-knowledge: The verifier $\mathcal{V}$ gains no additional knowledge beyond the validity of statements during the protocol execution, (II) Simulation sound: It is computationally infeasible for any prover $\mathcal{P}$ to persuade the verifier $\mathcal{V}$ of the validity of an invalid statement (selected by $\mathcal{P}$), even after having seen a polynomial number of simulated proofs for their chosen statements.

A.4 Oracle Systems

Privacy-Preserving Oracles [ZCC+16, ZMM+20], leverage cryptographic techniques that enable a user ($\mathcal{P}$) to prove to a verifier ($\mathcal{V}$) arbitrary statements about their data, which is securely retrieved from an authoritative web server ($\mathcal{S}$). This process ensures data privacy is preserved throughout. For instance, $\mathcal{P}$ can provide an oracle proof $\pi^{\mathcal{O}}$ to convince $\mathcal{V}$ that she is over 18, based on information sourced from her online U.S. State Department account ($\mathcal{S}$) without revealing any additional data beyond this statement. Oracles play a pivotal role in granting a wide range of applications– both in blockchain and traditional systems– access to private and public web data. Notable examples include credential creation from legacy data in DID systems and feeding real-world data to privacy-preserving smart contracts in Decentralized Finance (DeFi) [W3C18, BCC+21].

B Proofs

B.1 Security Analysis of tACT

In this section we present the security proofs of our tACT construction.

Theorem 2. *[Unforgeability] If tSW provides unforgeability, the cut-and-choose scheme is sound, and the ZKPoK argument is knowledge sound, then, the tACT scheme in Algorithm 1 satisfies unforgeability from Definition 3.*

Proof. Assume there is an adversary $\mathcal{A}$ that wins the unforgeability security game in Figure 4 with non-negligible probability. We construct three PPT adversaries: $\mathcal{B}_{\mathsf{tSW}}$ which breaks the unforgeability of the tSW, $\mathcal{B}_{\mathsf{SND}}$ which breaks the soundness of the cut-and-choose scheme, and $\mathcal{B}_{\mathsf{zkSND}}$ breaks the soundness of the zero-knowledge argument. We show that:

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{tACT},\mathcal{A}} \leq (T+L+1)\cdot \mathsf{Adv}^{\mathsf{uf}}_{\mathsf{tSW},\mathcal{B}_{\mathsf{tSW}}}(\lambda) + T\cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{SND},\mathcal{B}_{\mathsf{SND}}}(\lambda)$$

$$+T\cdot \mathsf{Adv}^{\mathsf{zksound}}_{\mathsf{SND},\mathcal{B}_{\mathsf{zkSND}}}(\lambda). \tag{1}$$

Construction of the adversaries. Let us first construct $\mathcal{B}_{\mathsf{tSW}}$. $\mathcal{B}_{\mathsf{tSW}}$ obtains pk from the challenges for the tSW scheme and provides it as public parameters to $\mathcal{A}$. $\mathcal{B}_{\mathsf{tSW}}$ answers signing queries $\pi^* = \{\mathsf{cm}, \{\mathsf{cm}_k\}_{k\in[1,n]}, \mathsf{cm}_{\mathsf{k}}\}$ from $\mathcal{A}$ as follows. If π^* fails to verify, abort. Else, $\mathcal{A}$ queries the tSW challenger for signatures and returns the result. We consider the output $\mathsf{h} := (\mathsf{token}_i := s_i, \pi_{tok_i})_{i\in[T+1]}$ that $\mathcal{A}$ generates. We write $\pi_{tok,i} = (\{s'_{k,i}\}_{k\in[1,n]}, C_i, \{R'_{k,i}\}_{k\in C_i}, \mathsf{pk}', \pi_{\mathsf{zk}})$. At the end, $\mathcal{B}_{\mathsf{tSW}}$ does the following:

- If $\mathcal{A}$ produced a Type-1 forgery, it uniformly at random chooses $i^* \in [T+1]$ and return the ith token $\pi_{tok_{i^*}} = (\{s'_{k,i^*}\}_{k\in[1,n]}, C_{i^*}, \{R'_{k,i^*}\}_{k\in C_{i^*}}, \mathsf{pk}', \pi_{\mathsf{zk}})$ to the tSW challenger as its forgery.
- If $\mathcal{A}$ produced a Type-2 forgery with set $\tau \subset [T]$. Assume without loss of generality that $|\tau| = L+1$. $\mathcal{B}_{\mathsf{tSW}}$ select uniformly at random $i^* \in \tau$ and outputs the i^*-th token $\pi_{tok_{i^*}} = (\{s'_{k,i^*}\}_{k\in[1,n]}, C_{i^*}, \{R'_{k,i^*}\}_{k\in C_{i^*}}, \mathsf{pk}', \pi_{\mathsf{zk}})$ to the tSW challenger as its forgery.

Now, let us construct $\mathcal{B}_{\mathsf{SND}}$. $\mathcal{B}_{\mathsf{SND}}$ runs the cut-and-choose challenger and does the following:

- If $\mathcal{A}$ produced a Type-1 forgery, it outputs $\perp$ to the cut-and-choose soundness challenger.
- If $\mathcal{A}$ produced a Type-2 forgery with set $\tau \subset [T]$. Assume without loss of generality that $|\tau| = L+1$. $\mathcal{B}_{\mathsf{SND}}$ select uniformly at random $j^* \in [T]$ and outputs the j^*-th query $\pi^* = (\mathsf{cm}_{j^*}, \{\mathsf{cm}_{k,j^*}\}_{k\in[1,n]}, \mathsf{cm}_{\mathsf{k}})$ made by $\mathcal{A}$ to tACT.tIssue to the cut-and-choose challenger (as a proof of an invalid statement).

Now, we construct $\mathcal{B}_{\mathsf{zkSND}}$. $\mathcal{B}_{\mathsf{zkSND}}$ runs the ZK soundness challenger and does the following:

- If $\mathcal{A}$ produced a Type-1 forgery, it outputs $\perp$ to the ZK soundness challenger.
- If $\mathcal{A}$ produced a Type-2 forgery with set $\tau \subset [T]$. Assume without loss of generality that $|\tau| = L{+}1$. $\mathcal{B}_{\mathsf{zkSND}}$ select uniformly at random $j^* \in [T]$ and outputs the j^*-th query $\pi^*_{\mathsf{token}} = \{\{s'_{k,j^*}\}_{k\in[1,n]}, C_{j^*}, \{R'_{k,j^*}\}_{k\in C}, \pi_{zk,j^*}, \mathsf{cm}_{j^*}, \mathsf{cm}_{\mathsf{k},j^*}, \mathsf{pk}'_{j^*}\}$ made by $\mathcal{A}$ to $\mathsf{tACT.Verify}$ to the ZK soundness challenger (as a proof of an invalid statement).

Remark that $\mathcal{B}_{\mathsf{tSW}}$, $\mathcal{B}_{\mathsf{SND}}$, and $\mathcal{B}_{\mathsf{zkSND}}$ cannot know (in polynomial time) whether the output to their tSW/cut-and-choose/ZK challenger is valid, i.e., if they will win the $\mathsf{tSW}_{\mathcal{B}_{\mathsf{tSW}}}(\lambda)$, $\mathsf{SND}_{\mathcal{B}_{\mathsf{SND}}}(\lambda)$, and $\mathsf{zkSND}_{\mathcal{B}_{\mathsf{zkSND}}}(\lambda)$ games. However, this is not an issue to prove that the advantage of at least one of those two adversaries is non-negligible if the advantage of $\mathcal{A}$ in unforgability experiment is non-negligible. . Finally, in order to extract r_i and a_i (following the experiment of unforgability of tACT section 3.1), we run the extractor Ext of the underlying ZKPoK's extractor to extract the committed values r_i and a_i.

Our proof strategy is the following. We define events $\mathsf{E}_{\mathsf{1tSW},i}$, $\mathsf{E}_{\mathsf{2tSW},k}$, $\mathsf{E}_{\mathsf{2,SND},j}$, and $\mathsf{E}_{\mathsf{2,zkSND},j}$ for $i \in [T+1]$, $k \in [L]$, $j \in [T]$ so that:

- Exactly one of these events must happen if the adversary wins.
- Conditioned on any of these events, either $\mathcal{B}_{\mathsf{tSW}}$ or $\mathcal{B}_{\mathsf{SND}}$, or $\mathcal{B}_{\mathsf{zkND}}$ wins EUF-CMA or soundness respectively.

Note that there is no need for these events to be efficiently checked for the proof to go through.

Definition of Events. Let us first define events $\mathsf{E}_{\mathsf{1tSW},i}$ that are associated with a Type-1 forgery. In that case, $\mathcal{A}$ generated $T+1$ tokens by doing T signature queries. $\mathcal{A}$ queries Sign only T times and there are $T+1$ valid tokens $\{\pi_{tok_i}\}_{i\in[T+1]}$, there exists $i \in [T+1]$ so that π_{tok_i} (and Token_i) is not in the query's list queried to Sign. And π_{tok_i} is a forgery for the tSW scheme.

We define $\mathsf{E}_{\mathsf{1tSW},i}$ the event that $\mathcal{A}$ produced a Type-1 forgery and π_{tok_i} (and Token_i) is not in the query's list queried to Sign, and no event $\mathsf{E}_{\mathsf{1tSW},i'}$ happened for $i' < i$. (This last constraint is to ensure that the events $\mathsf{E}_{\mathsf{1tSW},i}$ are disjoint.) Equivalently, $\mathsf{E}_{\mathsf{1tSW},i}$ is defined as the event that $\mathcal{A}$ produced a Type-1 forgery and π_{tok_i} if the $\pi_{tok_{i'}}$ that is not in the query's list queried to Sign.

Let us now consider Type-2 forgeries. Let us show that at least one of the following events must happen:

- Event $\mathsf{E}_{\mathsf{2tSW},k}$: $\mathcal{A}$ made a Type-2 forgery (and not a Type-1 forgery) for a set $\tau = i_1, \ldots, i_L$ with $i_1 < \cdots < i_L$, so that $\pi_{tok_{i_k}}$ is not in the query's list queried to Sign, and so that no event $\mathsf{E}_{\mathsf{2tSW},k'}$ happened for $k' \leq k$.
- Event $\mathsf{E}_{\mathsf{2,SND},i}$:
  - $\mathcal{A}$ made a Type-2 forgery (and not a Type-1 forgery) and,

- its i-th signing query is $(\mathsf{Token}_i, \pi_{tok_i})$ where π_{tok_i} is valid but proves a wrong cut-and-choose statement on the Token_i, and,
- neither event $\mathsf{E}_{2\mathsf{tSW},k}$ nor $\mathsf{E}_{2,\mathsf{SND},j}$ happened for $j' < j$ and any k.

The event $\mathsf{E}_{2,\mathsf{zkSND},i}$ is similar to $\mathsf{E}_{2,\mathsf{SND},i}$ but there π_{tok_i} is valid but proves a wrong ZK statement on the Token_i, and, non of the events $\mathsf{E}_{2\mathsf{tSW},k}$ or $\mathsf{E}_{2,\mathsf{SND},j}$ or $\mathsf{E}_{2,\mathsf{zkSND},j}$ happened for $j' < j$ and any k.

Let us assume by contradiction that none of the events happened (assuming $\mathcal{A}$ made a Type-2 forgery). Then since no $\mathsf{E}_{2\mathsf{tSW},k}$ happened, each $\pi_{tok_{i_k}}$ is in the query's list queried to Sign. Since there are $L+1$ of them and there are only L clients, at least two of them were made for the same client. These two queries are for the same message on which the Type-2 forgery was made. But since the cut-and-choose proofs were all proving valid statements (as no event $\mathsf{E}_{2,\mathsf{SND},i}$ occurred), this is a contradiction. So at least one of the events must happen.

Conclusion of the Proof. Since at least one of the events $\mathsf{E}_{1\mathsf{tSW},i}$, $\mathsf{E}_{2\mathsf{tSW},k}$, and $\mathsf{E}_{2,\mathsf{SND},i}$ must happen and these events are disjoint, we have that from the total probability rule:

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{tACT},\mathcal{A}} = \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1] = \sum_{i=1}^{T+1} \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}_{1\mathsf{tSW},i}]$$

$$+ \sum_{k=1}^{L} \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}_{2\mathsf{tSW},j}] + \sum_{j=1}^{T} \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}_{2\mathsf{tSW},j}]$$

In addition, looking at the definition of $\mathcal{B}_{\mathsf{tSW}}$ and from the fact this reduction perfectly simulates the oracles for $\mathcal{A}$:

$$\sum_{i=1}^{T+1} \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}_{1\mathsf{tSW},i}]$$

$$= \sum_{i=1}^{T+1} \mathsf{Pr}[\mathsf{uf}_{\mathcal{B}_{\mathsf{tSW}}}(\lambda) = 1 \wedge \mathsf{Event}_{1\mathsf{tSW},i} | i^* = i]$$

$$= \mathsf{Pr}[i^* = i]^{-1} \sum_{i=1}^{T+1} \mathsf{Pr}[\mathsf{uf}_{\mathcal{B}_{\mathsf{tSW}}}(\lambda) = 1 \wedge \mathsf{E}_{1\mathsf{tSW},i} \wedge i^* = i]$$

$$\leq \mathsf{Pr}[i^* = i]^{-1} \sum_{i=1}^{T+1} \mathsf{Pr}[\mathsf{uf}_{\mathcal{B}_{\mathsf{tSW}}}(\lambda) = 1 \wedge i^* = i]$$

$$= (T+1) \cdot \mathsf{Pr}[\mathsf{uf}_{\mathcal{B}_{\mathsf{tSW}}}(\lambda) = 1].$$

Similarly, we have,

$$\sum_{k=1}^{L} \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}_{2\mathsf{tSW},k}] \leq L \cdot \mathsf{Pr}[\mathsf{uf}_{\mathcal{B}_{\mathsf{tSW}}}(\lambda) = 1],$$

$$\sum_{j=1}^{T} \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}_{\mathsf{2SND},j}] \leq T \cdot \mathsf{Pr}[\mathsf{SND}_{\mathcal{B}_{\mathsf{SND}}}(\lambda) = 1].$$

$$\sum_{j=1}^{T} \mathsf{Pr}[\mathsf{uf}_{\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}_{\mathsf{2zkSND},j}] \leq T \cdot \mathsf{Pr}[\mathsf{zkSND}_{\mathcal{B}_{\mathsf{zkSND}}}(\lambda) = 1].$$

Together this concludes Equation (1).

Theorem 3. *[Unlinkability] If the* Ped.Commit *scheme provides hiding property, the ZKPoK scheme provides zero-knowledge property, and H is modeled as a random oracle, then the tACT scheme in Algorithm 1 satisfies unlinkability from Definition 4.*

Proof. We conduct the proof using hybrid games:

Game$_0$: This game represents the unlinkability security game.

Game$_1$: This game is similar to the previous one, except now all the cut-and-choose proofs are simulated.

Game$_0$-Game$_1$: This hybrid is computationally indistinguishable from the previous one due to the hiding property of the cut-and-choose argument.

Game$_2$: This game is similar to the previous one except that in any Unblind computation, instead of computing $C \leftarrow \mathsf{H}'(\mathsf{Token})$, it chooses C uniformly at random.

Game$_1$-Game$_2$: This game is indistinguishable from the previous one assuming H' is modeled as a random oracle.

Game$_3$: This game is similar to the previous one, except now all the ZK proofs are simulated.

Game$_2$-Game$_3$: This hybrid is computationally indistinguishable from the previous one due to the zero-knowledge property of the ZK argument.

Game$_4$: This game is similar to the previous one except that for every token request and associated unblind query, a is chosen uniformly at random.

Game$_3$-Game$_4$: This game is indistinguishable from the previous one assuming Ped.Commit scheme provides the hiding property.

In the final hybrid, the outcome of any token request is unrelated to the inputs a. The result of an unblind request solely comprises the underlying message and remains independent to the output of the corresponding token request. In this last game, no component from the computation of the token request is utilized during unblinding, except for the message itself. Consequently, the adversary's advantage is 0, leading to the conclusion of the proof.

B.2 Security Analysis of S3ID

Now, we provide the security proofs of the S3ID scheme.

Theorem 4. *[Unforgeability] If tSW provides unforgeability security and ZKPoK is sound, then, the S3ID scheme in Algorithm 2 satisfies unforgeability from Definition 6.*

Proof. The proof of unforgeability for $\mathsf{S3ID}$ is similar to the proof in Theorem 2. It relies on the soundness property of the ZKPoK and the unforgability of the tSW scheme. Specifically, the adversary attempting to forge is faced with the following possibilities:

- Computing an invalid signature ζ (while computing a valid τ and its corresponding ZKPoK proof π).
- Computing an invalid τ along with a proof π that passes the verification (while computing a valid signature ζ).

It is evident that the adversary cannot succeed in either case without breaking the unforgeability of the tSW scheme in the first scenario or violating the soundness property of the ZKPoK in the latter case.

Theorem 5. *[Sybil-resistance] If tACT provides unforgeability security and ZKPoK is sound, then, the S3ID scheme in Algorithm 2 satisfies Sybil-resistance from Definition 7.*

Proof. This is a direct result of the unforgeability of the tACT scheme proved in Theorem 2.

Theorem 6. *[Strong Unlinkability] If the* $\mathsf{Ped.Commit}$ *scheme provides the hiding property, H is modeled as a random oracle, ZKPoK provides zero-knowledge property, and the tACT scheme is unlikable, then the S3ID scheme in Algorithm 2 satisfies strong unlinkability from Definition 8.*

Proof. We proceed with the proof using hybrid games:

Game$_0$: This game is the unlinkability game in Figure 8.

Game$_1$: This game is similar to the previous one except now all the ZKPoK proofs are simulated.

Game$_0$-Game$_1$: This hybrid is computationally indistinguishable from the previous one due to the zero-knowledge property of the ZKPoK argument.

Game$_2$: This game is similar to the previous one except that in any $\mathsf{AppCred}$ computation, it chooses τ uniformly at random instead of computing $\tau = g^r \cdot u_0^{|\mathbb{Q}|\mathsf{k}} \cdot \prod_{i \in \mathbb{Q}} u_i^{\mathsf{a}_i}$.

Game$_1$-Game$_2$: This game is indistinguishable from the previous one assuming the hiding property of the commitment τ and the zero-knowledge property of the ZKPoK.

Game$_3$: This game is similar to the previous one except that in any $\mathsf{AppCred}$ computation, instead of computing $\mathsf{tg} = \mathsf{PRF}(\mathsf{k}, l)$, it chooses tg uniformly at random.

Game_2-Game_3: This game is indistinguishable from the previous one assuming PRF scheme.

Game$_4$: This game is similar to the previous one except that for every token request and associated unblind query, a is chosen uniformly at random.

Game_3-Game_4: This game is indistinguishable from the previous one assuming Ped.Commit scheme provides the hiding property.

In the last hybrid, token requests become independent of a. Unblind request outputs solely include the underlying message and are unrelated to associated token request outputs. Consequently, the adversary's advantage is 0, concluding the proof.