Fraud Mitigation in Privacy-Preserving Attribution

Rutchathon Chairattana-Apirom
University of Washington

Stefano Tessaro
University of Washington

Nirvan Tyagi
University of Washington

Abstract. Privacy-preserving advertisement attribution allows websites selling goods to learn statistics on which advertisement campaigns can be attributed to converting sales. Existing proposals rely on users to locally store advertisement history on their browser and report attribution measurements to an aggregation service (instantiated with multiparty computation over non-colluding servers). The service computes and reveals the aggregate statistic. The service hides individual user contributions, but it does not guarantee integrity against misbehaving users that may submit fraudulent measurements. Our work proposes a new cryptographic primitive, *secret share attestation*, in which secret shares input into a multiparty computation protocol are accompanied by an attestation of integrity by a third party: advertisers include signature attestations when serving ads that are later included in contributed measurements. We propose two constructions based on the standards-track BBS signatures and efficient signatures over equivalence classes, respectively. We implement and evaluate our protocols in the context of the advertising application to demonstrate their practicality.

Contents

1 Introduction

There are many scenarios in which data collection is necessary for the functioning of a service, naturally introducing a trade-off between user privacy and utility. *Digital advertising* is perhaps one of the most prominent examples. On the one hand, it is essential for sustaining a wide range of services. On the other, it raises significant privacy concerns. The one particular concern we focus on here is user tracking in the context of *attribution*, which seeks to link *impressions* on users during their online browsing (e.g., viewing a particular ad) to *conversions*, i.e., subsequent outcomes for that user (e.g., purchasing an item). Advertisers wish to assign monetary value to these impressions based on conversion outcomes. To achieve this effectively, it is helpful to aggregate conversion data across many users in order to evaluate the efficacy of individual ad campaigns. Attribution's use cases extend beyond advertisements—for example, a website could carry out an A/B test to see which type of contents leads to the highest user engagement.

It is therefore not surprising that several industry players, under the umbrella of the W3C's Private Advertising Technology Community Group (PATCG), are developing an interoperable privacy-preserving attribution (PPA) architecture [PLS+25], which consolidates and generalizes earlier proposals such as Google's ARA [HDRL25], Meta and Mozilla's IPA [TTS+23], and Apple's PAM [Win23]. These architectures fundamentally rely on the advertiser requesting an *encrypted* attribution report from a user. This report contains sensitive browsing information and can only be decrypted by an aggregation service. Such a service, in turn, may rely either on trusted hardware computation [HDRL25] or on split-trust multiparty computation (MPC) among non-colluding servers [TTS+23, Win23, GPP+25]. In the latter case, in particular, the reports are secret shared first, and the individual shares are encrypted to the corresponding MPC parties.

THE CHALLENGE OF FRAUD MITIGATION. A concern is that the integrity of attributions is affected by fraudulent reports. However, the PPA architecture described above does not attempt to prevent *impression fraud*, where a conversion report includes impressions that never actually occurred. This issue was addressed in earlier proposals, such as Apple's Private Click Measurement (PCM) [Wil21b, Wil21a] and Facebook's Deidentified Authenticated Telemetry (DIT) [HIJ+21, IT19]. In these systems, the validity of impressions is attested by an intermediary through a signature, with a signed record stored in the user's browser and later included in the final attribution report. During aggregation, reports with invalid signatures are discarded, and to preserve unlinkability between a report and the corresponding impression, a *blind* signature is employed.

To ensure signatures can be verified, PCM and DIT do not encrypt individual reports, and they are sent to the advertiser through an unauthenticated channel, thereby offering weaker privacy guarantees for users. In contrast, PPA only reveals the final aggregated statistics of these reports, but incorporating signature verification as part of the MPC protocol for MPC-based aggregation is fairly expensive and would add significant overhead to the proposed PPA protocol. [1] Consequently, PPA does not currently incorporate any fraud-detection mechanism.

OUR CONTRIBUTION. This work introduces a lightweight approach that integrates fraud detection into PPA based on split-trust MPC.

Our solution is fully transparent to the underlying aggregation MPC protocol (i.e., no changes to the MPC protocol itself are required), and relies on a new cryptographic primitive called *secret share attestation* (SSA). Instead of the intermediary simply blindly signing a value v, as in DIT and PCM, the user and intermediary engage in a protocol that produces a credential σ bound to v. When asked for a report by the conversion site, the user uses σ to generate secret shares $(\mathsf{ss}_i)_{i\in[n]}$ of v, which are encrypted to each of the aggregation servers, as well as some verification information. In turn, this consists of public verification information τ for the conversion site, and local verification information $(\tau_i)_{i\in[n]}$ for the aggregation servers (the latter are encrypted along with the shares). Usually, local verification is cheaper than global verification.

The validity of each share ss_i is ensured as long as τ is verified by the conversion site and each aggregation server verifies τ_i. In this case, each server uses ss_i as input to an existing aggregation MPC protocol. Crucially, no collection $\tau, (\mathsf{ss}_i, \tau_i)_{i\in[n]}$ can be generated without a valid credential σ for the value v reconstructed from the shares. Further, it cannot be non-trivially linked to the interaction with the intermediary that generated it.

[1] While the blind signature can be revealed to each participating party, the value that has been signed is provided only in secret-shared form.

We provide formal security definitions for SSA, along with two constructions. The first relies on BBS credentials [BBS04, ASM06, CDL16, TZ23, LKWL24] and the second is based on rerandomizable signatures over equivalence classes [FHS19, CL19, BF20]. The former provides a more efficient issuance protocol while the latter performs more efficiently at verification time. We benchmark both approaches and compare them against relevant baselines. We find that SSA opens up a new attractive deployment model for PPA in which input verification costs are shifted from the aggregation servers to the advertiser.

PRIOR WORKS: CERTIFIED INPUTS IN MPC. We think of SSA as being used in a context where user reports can be malicious, but the conversion site and the aggregation servers are honest-but-curious. This corresponds to the setting considered by the PPA standardization. This also enables a lightweight solution on top of existing PPA aggregation protocols without changing them.

In a contrast, a number of prior works [Bau16, BB16, KMW16, ZBB17, BJ18, ADEO21, DGPS24] consider the question of certifying/validating the inputs to MPC protocols with stronger security guarantees while however also modify the underlying MPC protocol. The earlier works of [Bau16, BB16, KMW16, ZBB17] specifically considered certifying inputs in two-party computations protocols that rely on garbled circuits. Blanton and Jeong [BJ18] applied CL signatures [CL03] to preserve unlinkability of the certified inputs in MPC protocols. Aranha et al.'s more efficient approach [ADEO21] is based on a protocol to efficiently compute bilinear pairing over secret shares, which allows for efficient verification of PS signatures [PS16] over a secret shared message. Most closely related to our work, Dutta et al. [DGPS24] study "authenticated MPCs" where the (possibly malicious) inputs and secret shares used in the protocol are certified by a central authority. They give a generic compiler based on BBS signatures [BBS04, ASM06, CDL16, TZ23] in which shares are verified through an interactive distributed proof of BBS opening among MPC servers (see Section 8 and Section 9 for further comparison).

2 Overview

This section provides an overview of privacy-preserving attribution, existing solutions for privacy-preserving fraud detection, and finally of our approach for integrating both. Figure 1 also serves as a summary of these architectures.

2.1 Privacy-Preserving Attribution

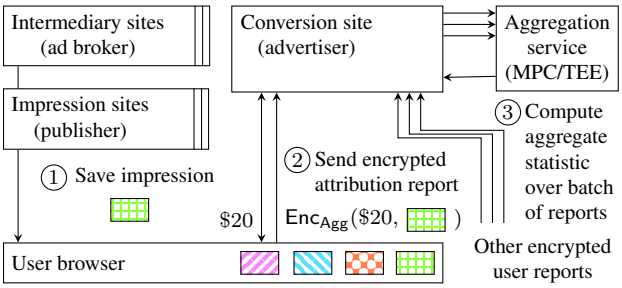
Attribution aims to identify and attribute *impressions* on users during online browsing to converted *outcomes* for that user, allocating value to those impressions based on the outcome. Websites are interested in aggregating data on these conversion outcomes across many users to understand which impressions are most effective. More concretely, consider the following two scenarios:

Advertising: A shopping website runs a number of ad campaigns targeting users with different interests, e.g., one ad campaign displays on gaming websites while another displays on cooking websites. Relevant impressions for users include being shown an ad and clicking on an ad on a campaign-targeted website. Conversion outcomes of increasing value might include adding an item to the shopping cart, subscribing to a mailing list, or completing a purchase on the host shopping website. The shopping website wants to learn which ad campaigns are most effective.

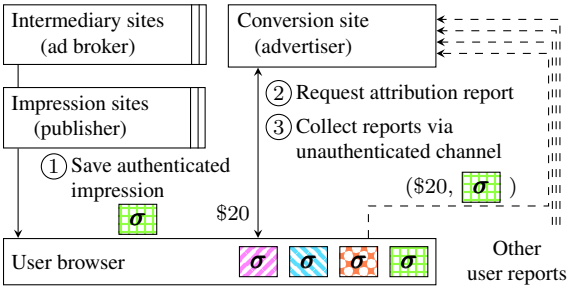
A/B UX Testing: A website wants to test a new tutorial user experience flow for users by running an A/B test with different tutorials. An impression for a user is the tutorial they are shown during onboarding. Conversion outcomes include user engagement with site features after the tutorial, e.g., failed workflows might result in negative value and completed workflows result in positive value. The website wants to learn which tutorial experience results in better outcomes.

We will use the advertising scenario as a running example throughout the text.

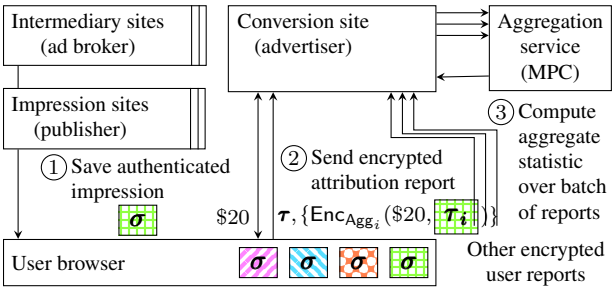
The architecture behind Privacy-Preserving Attribution Level 1 proposal (PPA) [PLS+25] by the W3C's Private Advertising Technology Community Group (PATCG) is illustrated in Figure 1a. It provides a browser API for *impression sites* to record impressions to be stored locally on the user's browser. Often times, the impression is recorded via a



(a) PPA, IPA, PAM: Private aggregation of attribution reports but no mitigation for impression fraud.



(b) PCM, DIT: Blind signatures for fraud mitigation but no private aggregation service.



(c) This work: Private aggregation and fraud mitigation.

Figure 1: Proposed architectures for privacy-preserving attribution. In the first step, user browsers save impressions served to them by sites during normal browsing activity. In the second step, a user and conversion site engage in a conversion activity of value $20, and the conversion site requests an attribution report. The proposed architectures differ in how this report is collected and aggregated. In (1c), the public τ is verified by the conversion site, and the local τ_i is encrypted to and verified by each aggregation server.

script loaded on the impression site from an *intermediary site*. For example, in the advertising scenario, an advertising broker company will play the role of an intermediary site hired by the shopping site. The gaming sites and cooking sites (ad publishers) load scripts from the ad broker site that display ads and record impressions. This distinction between impression site and intermediary site will be important during our discussion of fraud shortly.

Later, when a user achieves a conversion outcome on the *conversion site* (e.g., the shopping site), the conversion site can call the browser API to request a browser-generated *attribution report*. The conversion site provides a conversion value and *attribution function* for the browser to apply locally on stored user impressions (designated for the conversion site). The output report is a list of impressions with assigned numerical values. For example, the "last touch" attribution function attributes all of the conversion value to the most recent impression [CJK+23].

Importantly, the user attribution report is not given to the conversion site in the clear, as it contains private information about the user browsing history. Instead, the conversion site specifies a supported *aggregation service*. The browser encrypts the attribution report to the aggregation service and sends the encrypted report to the conversion site. The conversion site collects a set of encrypted reports over time and submits the set to the aggregation service which decrypts, performs aggregation, and returns the aggregate statistic to the conversion site. Proposed architectures for

privacy-preserving aggregation services include trusted hardware computation [HDRL25] and split-trust multiparty computation (MPC) among non-colluding servers [TTS+23, Win23, GPP+25]. In the trusted hardware service, reports are encrypted directly to the enclave via attestation. In the split-trust MPC service, reports are secret-shared locally and the report shares are encrypted to each MPC party. See Figure 1a for a depiction of this flow.

The privacy goals of the system dictate that as long as the aggregation service trust assumption holds (i.e., the trusted hardware is not compromised or the non-colluding MPC assumption holds) then even if impression, intermediary, and conversion sites collude, no user information should be revealed beyond what is revealed by the aggregate statistic. Lastly, differential privacy can be layered on top to provide additional individual privacy with respect to the revealed aggregate statistic [TKM+24]; random noise is added to the statistic by the aggregation service and the user tracks a privacy budget locally for each of their impressions.

2.2 Fraud Mitigation via Blind Signatures

PPA still allows users/browsers to submit fraudulent attribution reports enabling manipulation of the results of the attribution statistic, possibly influencing future actions taken by the conversion site. For example, from the advertising scenario, the publisher impression site serving ads as part of a campaign may want to artificially boost the value of its ads so that the conversion shopping site will pay for more ads in the future. It can hire a clickfarm or botnet to trigger conversion outcomes on the shopping site, e.g., by adding items to the shopping cart, and send fraudulent attribution reports that attribute large fake conversion values to a fake ad impression on the gaming site.

The first concern of fake conversion values can largely be addressed by aggregation services. The conversion site specifies an expected maximum conversion value sum for impressions in a report; recall, the conversion site requests the report after observing a conversion outcome of some value. A trusted hardware aggregation service checks this constraint directly after decrypting the report. A split-trust MPC aggregation service verifies lightweight cryptographic proofs of knowledge provided by the browser [CB17, BBC+19, YW22, BBC+23, RZCP24, ROCT24].

The second concern of fake impressions is unaddressed in the current PPA proposal. Apple's Private Click Measurement (PCM) [Wil21b, Wil21a] and Facebook's Deidentified Authenticated Telemetry (DIT) [HIJ+21, IT19] observe the ad broker company (intermediary) is aligned in incentive with its customers (conversion sites) to prevent fraud by impression sites in order to maintain its reputation as a trustworthy broker. Thus, the intermediary site can be trusted to attest to validity of impressions served on an impression site, e.g. by signing the impression with a digital signature that is stored with the impression locally on the user browser. When a conversion attribution report is generated, the browser includes the intermediary site signature with the attributed impression to prove that the impression is authentic. To address privacy, PCM and DIT propose the use of *blind signatures* [Cha82, DJW23]. A blind signature scheme allows for the browser to obtain a signature on the impression from the intermediary blindly. The intermediary does not learn the signature and cannot link the signature to the signing interaction when it is later revealed.

However, PCM and DIT do not use an aggregation service to collect reports. Instead, reports are collected through an unauthenticated channel directly to the conversion site. While the blind signature prevents linking via the signature, omitting an aggregation service means that individual unaggregated reports are revealed directly to the conversion site, which itself can be used to link users to browsing histories (see Figure 1b). These privacy concerns have led to PCM and DIT falling out of consideration for PATCG's standards search [Tho22]. As such, impression fraud is not addressed in the existing PPA proposal.

LIMITATIONS AND NON-GOALS. Impressions with valid signatures that are reported during conversion are guaranteed to be authentic. However, that guarantee does not eliminate fraud and reporting misbehavior:

Misattribution: Malicious reporters may not abide by the conversion site's requested attribution function, choosing to omit or include attribution to certain impressions.

Mix-and-match: Malicious reporters may collude with each other to share impressions and build an impression history that no single user ever experienced.

Inauthentic behavior: Botnets and clickfarms may go through a valid user flow (e.g., click an ad and sign up for a mailing list) and create a conversion report following the correct protocol.

Some of these limitations can be (partially) addressed by extending the basic protocol. Rate limiting [CHK+06, YW25] and expiration [TCR+22, HW25] can help with misattribution and mixing-and-matching. Binding impressions together across a "multi-touchpoint" user journey can help with mixing-and-matching and inauthentic behavior, e.g., by issuing impressions for acting like a normal user after clicking on an ad [IT19]. We discuss the compatibility of these extensions with our proposals and expand on open questions regarding limitations in Section 9.

2.3 Our Approach: Attestation of Secret Shares

We develop a new approach to compose signature attestation of impressions with private aggregation via split-trust MPC, as in IPA and PAM. We do not consider the case of aggregation via trusted hardware [CLPP25], for which integrating signature verification is quite simple, but for which side channels [ZSS+16, GESM17, BMW+18, LZW+21] can potentially degrade privacy. (In fact, blind signatures are not needed for report unlinkability; regular signatures suffice if the hardware is trusted not to leak its inputs.)

Simply providing a blind signature for the impression to each MPC server is not very efficient, as they only hold a share of the impression, and thus the verification algorithm needs to be run within the MPC, adding extra cost to the relatively simple MPC considered by PPA. Instead of a signature on the impression, in our approach, the intermediary gives the user a credential that enables generating verification information for each of the secret shares of the impression. When shares are sent to the MPC servers, the servers verify the verification information on their share; no non-native MPC is required–in particular, verification does not require interactions between the MPC servers. To prevent linkability to the signing event, the verification information will still need to be blind. Further, the secret shares themselves (the signed message) that are provided to the MPC servers should not be linkable to the signing event either. We formalize these requirements under a new cryptographic primitive that we call *secret share attestation* (Section 4) and propose two constructions (Sections 5 and 7) where the unlinkability is unconditional and does not require any cryptographic assumptions. The overall architecture is illustrated in Figure 1c.

2.3.1 Approach 1: BBS Signatures

Our first approach can be formulated generically from anonymous credentials [CL03, CL04]. Our instantiation builds on top of the BBS signature scheme [BBS04, ASM06, CDL16, TZ23] that is currently being considered for standardization [LKWL24] and which allows an efficient instantiation. Consider a pair of groups $\mathbb{G}, \widehat{\mathbb{G}}$ of prime order p with generators $G, \widehat{G}$ that admit a bilinear pairing e. Say the impression value to be reported is $v \in \mathbb{Z}_p$ and it is to be shared across an aggregation service with two servers. (Our complete constructions extend to support multiparty aggregation services (> 2), a threshold access structure, and vector-valued impressions.) Initially, the user obtains a BBS signature $\sigma = (A, e)$ on v from the intermediary.

$$A = \frac{1}{x+e}(G + vH)\,,$$

where $x \leftarrow\!\!\$\, \mathbb{Z}_p$ is the intermediary secret key, $e \leftarrow\!\!\$\, \mathbb{Z}_p$ is chosen at random, and H is generator of $\mathbb{G}$ with unknown discrete log. Note that with public key $\widehat{X} = x\widehat{G}$, signature σ can be verified using a pairing check.

Later, when a conversion site requests a report, the user picks random shares s_1, s_2 such that $s_1 + s_2 = v$, and commits to them as $C_i = r_i G + s_i H$ for $i = 1, 2$, where $r_1, r_2 \leftarrow\!\!\$\, \mathbb{Z}_p$. It also computes randomizations $\widetilde{A} = \alpha A$ and $\widetilde{B} = \alpha(G + vH - eA) = x\widetilde{A}$ for $\alpha \leftarrow\!\!\$\, \mathbb{Z}_p$. Then, the user sends to the conversion server:

- Encryptions of (s_i, r_i) under the respective public keys of the two aggregation servers.
- The *public verification information* τ consisting of $\widetilde{A}, \widetilde{B}, C_1, C_2$, as well as a non-interactive proof π for the following relation

$$\mathsf{R}_{\mathsf{Share}} := \left\{ \begin{array}{c} ((\widetilde{A}, \widetilde{B}, C_1, C_2); \\ (\alpha, e, v, s_1, s_2, r_1, r_2)) \end{array} : \begin{array}{l} \widetilde{B} = \alpha(G + vH) - e\widetilde{A} \wedge \\ v = s_1 + s_2 \ \wedge\ \forall i \in \{1,2\} : C_i = r_i G + s_i H \end{array} \right\}.$$

This proof is a Σ-protocol proof following the approach from [TZ23]. The conversion server checks that the proof is valid and that $\mathsf{e}(\widetilde{A}, \widehat{X}) = \mathsf{e}(\widetilde{B}, \widehat{G})$, and if so, forwards the ciphertexts to the respective aggregation servers, along with the corresponding commitment C_1 or C_2. Otherwise, the report is rejected.

Finally, aggregation server $i \in \{1,2\}$, given C_i and the decrypted s_i, r_i, accepts share s_i if $r_i G + s_i H = C_i$.

2.3.2 Approach 2: Equivalence Class Signatures

The prior construction offers an extremely lightweight interaction between the user and the intermediary, yet when the report contains a vector $\vec{v} \in \mathbb{Z}_p^m$, the added size of a report generally grows linearly in m, and so does the verification time by the conversion server. This is due to the included proof π.

Our second approach provides a different trade-off using ideas from rerandomizable signatures over equivalence classes [FHS19, CL19, BF20], which allow creating signatures on message *equivalence classes* such that the signature can be rerandomized *and* the message can be rerandomized to another member of the equivalence class. Concretely, we define an equivalence class for valid additive secret shares of a message and construct a rerandomizable signature scheme for it. To this end, consider encoding the secret shares within two Pedersen commitments:

$$C_1 = s_1 G + r_1 H\,, \quad C_2 = s_2 G + r_2 H$$

At a high level, our strategy will be for a user to acquire an equivalence class signature σ on (C_1, C_2) from the intermediary. For now, assume the user proves to the intermediary that C_1 and C_2 commit to s_1 and s_2 such that $s_1 + s_2 = v$; later we will show how to dispense with this proof. When including the impression in a report, the user will locally adapt (C_1, C_2) and σ to (C_1', C_2') and σ' such that C_1' and C_2' commit to a fresh secret sharing $s_1' + s_2' = v$. The conversion server verifies σ' against (C_1', C_2'). Further, the user opens C_1' to s_1' for the first aggregation server and C_2' to s_2' to the other.

As a starting point, consider the SoRC equivalence class signature scheme [BF20], in which an equivalence class is defined by a message $M \in \mathbb{G}$ and ElGamal public key $K \in \mathbb{G}$ and consists of all valid ElGamal ciphertexts of M:

$$[\![(K, M)]\!]_{\mathsf{ElG}} = \{(K, \gamma G, \gamma K + M) : \gamma \in \mathbb{Z}_p\}\,.$$

A minimal change to SoRC allows us to nearly achieve this goal by constructing a signature scheme for the equivalence class

$$[\![(C_1, C_2)]\!]_{\mathsf{SSInsecure}} = \{(C_1 + \gamma G, C_2 - \gamma G) : \gamma \in \mathbb{Z}_p\}\,.$$

This equivalence class supports adapting C_1 and C_2 to commit to $s_1 + \gamma$ and $s_2 - \gamma$, respectively, for any choice of $\gamma \in \mathbb{Z}_p$. Unfortunately, this approach is insecure for our application. Note that the commitment *message* has been updated and rerandomized by γ, but the commitment *randomness* was not changed. When a user adapts C_1 to C_1' by γ' and to C_1'' by γ'' for two different reports, the user opens C_1' and C_1'' to the aggregation server where the openings consist of $(s_1 + \gamma', r_1)$ and $(s_1 + \gamma'', r_1)$, respectively. Since randomness r_1 is unchanged between reports, the reports are trivially linkable.

We address this by additionally allowing the commitment randomness to be updated during adaption. However, by allowing arbitrary rerandomization of the commitments, the set of reachable group elements (C_1', C_2') no longer falls within distinct equivalence classes. Any pair of group elements are reachable!

$$[\![(C_1, C_2)]\!]_{\mathsf{SSNoEq}} = \{(C_1 + \gamma G + \beta_1 H, C_2 - \gamma G + \beta_2 H) : \gamma, \beta_1, \beta_2 \in \mathbb{Z}_p^3\}$$

Without distinct equivalence classes, it is not clear how to prove security notions such as unforgeability. We observe that in PPA we only care about adapting secret shared field elements. Instead of constructing an intermediate notion of equivalence classes over group elements, we construct a signature scheme directly for an equivalence class over (commitments to) *field elements.* A forgery is no longer defined by defining an equivalence class for the commitments (group elements). Rather, a forgery is defined by the ability *to open* a commitment to field elements outside the "field" equivalence class. Thus, we define the equivalence class of interest for refreshing secret shares as:

$$[\![v]\!]_{\mathsf{SS}} = \{(v + \gamma, -\gamma) : \gamma \in \mathbb{Z}_p\}\,.$$

Putting it all together, the following describes the equivalence class signature construction extended from SoRC for the SSInsecure equivalence class. The further changes needed to support rerandomizing the commitment randomness are included in blue. The intermediary creates the initial commitments using a canonical initial representation of impression value v, where commitment randomness r_1, r_2 and the second secret share s_2 are set to 0, and $s_1 = v$:

$$C_1 = vG + 0H\,, \quad C_2 = 0G + 0H\,.$$

Since our scheme will support rerandomizing the commitment and shares, it is okay for the intermediary to create the initial commitments; the user can randomize them later. As a result, no initial proof of knowledge is needed.

Assume again a pair of groups that admit a bilinear pairing. The intermediary holds the signature secret key $(x_1, x_2) \leftarrow\!\!\$\, \mathbb{Z}_p^2$ and publishes public key $(\widehat{X}_1 = x_1\widehat{G}, \widehat{X}_2 = x_2\widehat{G})$. The signature $\sigma = (Z, T, A, \widehat{A}, T_1, T_2)$ consists of:

$$\begin{aligned} Z &= \frac{1}{\alpha}(G + x_1 C_1 + x_2 C_2)\,, \\ T &= \frac{1}{\alpha}(x_1 G - x_2 G)\,, \quad A = \alpha G\,, \quad \widehat{A} = \alpha\widehat{G}\,. \\ T_1 &= \frac{1}{\alpha}(x_1 H)\,, \quad T_2 = \frac{1}{\alpha}(x_2 H) \end{aligned}$$

Verification occurs by checking the commitment openings and checking a few pairing product equations. The values T, T_1, T_2 constrain the way in which σ can be adapted. They are constructed to only allow refreshing the committed secret share (via T) and rerandomizing the commitments (via T_1, T_2). Given σ and (s_1, s_2, r_1, r_2), adaption to (s_1', s_2', r_1', r_2') and $\sigma' = (Z', A', \widehat{A}')$ is as follows:

$$\begin{aligned} &\gamma, \alpha' \leftarrow\!\!\$\, \mathbb{Z}_p\,, \quad \beta_1, \beta_2 \leftarrow\!\!\$\, \mathbb{Z}_p\,, \\ &s_1' \leftarrow s_1 + \gamma\,, \quad s_2' \leftarrow s_2 - \gamma\,, \\ &r_1' \leftarrow r_1 + \beta_1\,, \quad r_2' \leftarrow r_2 + \beta_2\,, \\ &C_1' \leftarrow C_1 + \gamma G + \beta_1 H\,, \quad C_2' \leftarrow C_2 - \gamma G + \beta_2 H\,, \\ &Z' \leftarrow \frac{1}{\alpha'}(Z + \gamma T + \beta_1 T_1 + \beta_2 T_2)\,, \\ &A' = \alpha' A\,, \quad \widehat{A}' = \alpha'\widehat{A}\,. \end{aligned}$$

For our purpose, the rerandomizations of T, T_1, and T_2 are not needed, as verifying Z' already guarantees authenticity of the rerandomized commitments C_1', C_2'. This results in shorter adapted signatures when we generalize to n parties.

Lastly, verification simply requires the user provide the commitments and commitment openings. The conversion server verifies the signature over the commitments, and the aggregation servers open the commitments given the opening. This procedure does not require any further proofs of knowledge as the commitments and openings (shares and randomness) are freshly randomized and can be given directly. In comparison to the BBS-based construction, the construction based on equivalence class signatures improves verification efficiency both in terms of bandwidth (constant proof size) and verifier computation by removing the need for extra zero-knowledge proofs. We give the definition and construction for the specific equivalence class signatures on commitments in Section 6.

3 Preliminaries

NOTATIONS. Denote λ as the security parameter, $[n, m] = \{n, n+1, \ldots, m\}$ for any integers n, m where $n \leqslant m$, and $[n] = [1, n]$ for any integer $n \geqslant 1$. Denote $y \leftarrow\!\!\$\, \mathsf{A}(x)$ as running a (probabilistic) algorithm A on input x with fresh randomness and $[\mathsf{A}(x)]$ as the set of possible outputs; $(y_1, y_2) \leftarrow\!\!\$ \langle \mathsf{A}(x_1) \rightleftharpoons \mathsf{B}(x_2) \rangle$ denotes a pair of interactive algorithms A, B with inputs x_1, x_2 and outputs y_1, y_2, resp. We will denote formal variables in polynomials with sans-serif letters (e.g., X, Y). For any prime modulus p, let $\mathbb{Z}_p[\mathsf{X}]$ denote the ring of polynomials containing $g(\mathsf{X}) = \sum_{i=0}^{d} a_i \mathsf{X}^i$ with coefficients in $\mathbb{Z}_p$ and with d as the degree of $g(\mathsf{X})$, alternatively denoted using $\deg g$. We might refer to polynomial $g(\mathsf{X})$ using the shorthand g when it is clear from the context. We often denote vectors using the overhead arrow (e.g., $\vec{v}, \vec{H}$).

Game q-SDH$^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)$:	Game (q_1,q_2)-DL$^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)$:
$\mathsf{pp} = (p,\mathbb{G},\widehat{\mathbb{G}},\mathbb{G}_T,\mathsf{e}) \leftarrow\!\$\ \mathsf{GGen}(1^\lambda)$	$\mathsf{pp} = (p,\mathbb{G},\widehat{\mathbb{G}},\mathbb{G}_T,\mathsf{e}) \leftarrow\!\$\ \mathsf{GGen}(1^\lambda)$
$G \leftarrow\!\$\ \mathbb{G}^*; \widehat{G} \leftarrow\!\$\ \widehat{\mathbb{G}}^*; x \leftarrow\!\$\ \mathbb{Z}_p$	$G \leftarrow\!\$\ \mathbb{G}^*; \widehat{G} \leftarrow\!\$\ \widehat{\mathbb{G}}^*; x \leftarrow\!\$\ \mathbb{Z}_p$
$(e,Z) \leftarrow\!\$\ \mathcal{A}(\mathsf{pp}, G, (x^i G)_{i\in[q]}, \widehat{G}, x\widehat{G})$	$x' \leftarrow\!\$\ \mathcal{A}(\mathsf{pp}, G, (x^i G)_{i\in[q_1]}, \widehat{G}, (x^i \widehat{G})_{i\in[q_2]})$
return $\left(Z = \frac{1}{x+e} G\right)$	**return** $(x = x')$

Figure 2: Games q-SDH and (q_1,q_2)-DL

Alg. BBS.Setup(1^λ) :	Alg. BBS.KG(pp) :	Alg. BBS.Ver(pp, pk, $\vec{m}$, σ) :
$(p,\mathbb{G},\widehat{\mathbb{G}},\mathbb{G}_T,\mathsf{e}) \leftarrow\!\$\ \mathsf{GGen}(1^\lambda)$	$x \leftarrow\!\$\ \mathbb{Z}_p$	**parse** $(A,e) \leftarrow \sigma$
$G \leftarrow\!\$\ \mathbb{G}^*; \widehat{G} \leftarrow\!\$\ \widehat{\mathbb{G}}^*; \vec{H} \leftarrow\!\$\ \mathbb{G}^\ell$	**return** $(\mathsf{sk} \leftarrow x, \mathsf{pk} \leftarrow x\widehat{G})$	$C \leftarrow G + \sum_{i=1}^{\ell} m_i H_i$
$\mathsf{pp} \leftarrow (p, G, \vec{H}, \widehat{G}, \mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, \mathsf{e})$	Alg. BBS.Sign(pp, sk, $\vec{m} \in \mathbb{Z}_p^\ell$) :	**return** $A \neq 1_{\mathbb{G}} \ \wedge\ \mathsf{e}(A, \mathsf{pk} + e\widehat{G}) = \mathsf{e}(C, \widehat{G})$
return pp	$e \leftarrow\!\$\ \mathbb{Z}_p; A \leftarrow \frac{1}{x+e}\left(G + \sum_{i=1}^{\ell} m_i H_i\right)$	
	return (A,e)	

Figure 3: The signature scheme $\mathsf{BBS} = \mathsf{BBS}[\mathsf{GGen},\ell]$ is parameterized by GGen and the message length $\ell = \ell(\lambda) \geqslant 1$.

BILINEAR PAIRING GROUPS. We work with groups $\mathbb{G}$ with prime-order p and additive notations. We denote $0_{\mathbb{G}}$ as the identity and $\mathbb{G}^*$ as the set of generators of $\mathbb{G}$. We write group elements and scalar with upper-case and lower-case letters, respectively. For $G \in \mathbb{G}^*, H \in \mathbb{G}$, $a = \mathrm{DL}_G(H) \in \mathbb{Z}_p$ if $H = aG$.

A *bilinear group parameter generator* is a probabilistic algorithm GGen taking as input 1^λ and outputting $(p,\mathbb{G},\widehat{\mathbb{G}},\mathbb{G}_T,\mathsf{e})$ such that $\mathbb{G},\widehat{\mathbb{G}}$ and $\mathbb{G}_T$ are groups of λ-bit prime order p, and $\mathsf{e} : \mathbb{G} \times \widehat{\mathbb{G}} \to \mathbb{G}_T$ is a bilinear map. The map e satisfies (1) *bilinearity*, i.e., for any $A \in \mathbb{G}, B \in \widehat{\mathbb{G}}$ and $x,y \in \mathbb{Z}_p$, $\mathsf{e}(xA, yB) = (xy)\cdot\mathsf{e}(A,B)$, and (2) *non-triviality*, i.e., for any $G \in \mathbb{G}^*, \widehat{G} \in \widehat{\mathbb{G}}^*$, $\mathsf{e}(G,\widehat{G}) \in \mathbb{G}_T^*$.

SECURITY ASSUMPTIONS. Our constructions will depend on a variant of the q-*Strong Diffie-Hellman* (q-SDH) assumption [BB08], in a format supporting type-3 pairings. We will also consider the (q_1,q_2)-*Discrete Logarithm* (q-DL) assumption [Lip12], a generalization of the q-DL assumption (implied by the q-SDH assumption and equivalent to the $(q,1)$-DL assumption). Note that we drop q_1,q_2 when $q_1 = q_2 = 1$ and refer to it as the *Discrete Logarithm* (DL) assumption. For any adversary $\mathcal{A}$ in the corresponding games q-SDH and (q_1,q_2)-DL (in Figure 2), we define its advantage as $\mathsf{Adv}^{(q\text{-sdh}/(q_1,q_2)\text{-dl})}_{\mathsf{GGen}}(\mathcal{A},\lambda) := \mathsf{Pr}[(q\text{-SDH}/(q_1,q_2)\text{-DL})^{\mathcal{A}}_{\mathsf{GGen}}(\lambda) = 1]$.

BBS SIGNATURES. We describe the $\mathsf{BBS} = \mathsf{BBS}[\mathsf{GGen},\ell]$ signature scheme in Figure 3 with notations adapted from [TZ23]. The scheme is parameterized by a bilinear group parameter generator GGen and an integer $\ell = \ell(\lambda) \geqslant 1$ denoting the message length (defining the message space $\mathsf{BBS.M} = \mathbb{Z}_p^\ell$). In the case that $x + e = 0$, we denote $1/0 = 0$ for ease of syntax.

RELATIONS AND NON-INTERACTIVE PROOFS. Let $\mathsf{R} \subseteq \mathcal{X} \times \mathcal{W}$ be a relation. A non-interactive zero-knowledge (NIZK) proof system Π for a relation R is a tuple of algorithms $(\Pi.\mathsf{Prove}^{\mathsf{H}}, \Pi.\mathsf{Ver}^{\mathsf{H}})$ with access to a random oracle $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$ for some challenge space $\mathcal{C}$ with the following syntax:

$\pi \leftarrow\!\$\ \Pi.\mathsf{Prove}^{\mathsf{H}}(x,w)$: for $(x,w) \in \mathsf{R}$ outputs a proof π.

$0/1 \leftarrow \Pi.\mathsf{Ver}^{\mathsf{H}}(x,\pi)$: verifies a proof π for statement x.

We require a NIZK to be correct, zero-knowledge, and straightline extractable knowledge-sound for a relaxed relation $\widetilde{\mathsf{R}} \supseteq \mathsf{R}$. These are defined as follows:

Correctness. For any $(x,w) \in \mathsf{R}$ and $\pi \leftarrow\!\$\ \Pi.\mathsf{Prove}^{\mathsf{H}}(x,w)$, $\Pi.\mathsf{Ver}^{\mathsf{H}}(x,\pi) = 1$ with probability 1, where the probability is over the random choice of H and the random coins of Π.Prove.

Zero-knowledge. There exists a simulator Sim which is allowed to reprogram H such that for any adversary $\mathcal{A}$ with bounded access to H, the following advantage is bounded:

$$\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathsf{Sim}}(\mathcal{A},\lambda) := \left|\mathsf{Pr}[\mathcal{A}^{\mathsf{H},\mathrm{O}_0}(1^\lambda) = 1] - \mathsf{Pr}[\mathcal{A}^{\mathsf{H},\mathrm{O}_1}(1^\lambda) = 1]\right| .$$

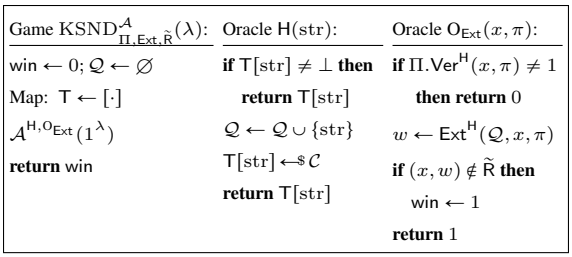


Figure 4: Straightline extractable knowledge soundness game for NIZK Π.

The oracles $\mathrm{O}_b(x,w)$ does the following: If $(x,w) \notin \mathsf{R}$ then return $\perp$. If $b=0$, then return $\pi \leftarrow\!\!\$\ \Pi.\mathsf{Prove}^{\mathsf{H}}(x,w)$. Otherwise, return $\pi \leftarrow\!\!\$\ \mathsf{Sim}^{\mathsf{H}}(x)$.

Relaxed knowledge-soundness. A Π is straight-line extractable knowledge-sound for a *relaxed relation* $\widetilde{\mathsf{R}} \supseteq \mathsf{R}$ if there exists an extractor Ext who has access to the adversary's random oracle queries such that for any adversary $\mathcal{A}$ playing the game KSND (defined in Figure 4), the following advantage is bounded

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\Pi,\mathsf{Ext},\widetilde{\mathsf{R}}}(\mathcal{A},\lambda) := \mathsf{Pr}[\mathrm{KSND}^{\mathcal{A}}_{\Pi,\mathsf{Ext},\widetilde{\mathsf{R}}}(\lambda)=1]\,.$$

We note that later on we will assume that our proof systems are straight-line extractable knowledge sound. This is justified in the Algebraic Group Model (AGM) [FKL18] with extractors that have access to the algebraic representations of the group elements that the adversary outputs.

4 Secret Share Attestation

In this section, we introduce Secret Share Attestation (SSA). SSAs allow users to obtain from the issuer a credential σ for a value v satisfying an issuance predicate ϕ. The user, using σ and a sharing predicate ϕ', may produce secret shares $(\mathsf{ss}_i)_{i\in[n]}$ of v and public and local verification information τ and $(\tau_i)_i$ such that, for each $i \in [n]$, $(\tau,\mathsf{ss}_i,\tau_i)$ attests that ss_i is a secret share of a value v certified by the issuer and $\phi'(v)=1$.

4.1 Syntax

A SSA scheme consists of the following algorithms. We describe each algorithm with respect to the intended application to PPA as illustrated in Figure 1c.

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SSA.Setup}(1^\lambda,t,n)$**:** generates public parameters depending on a threshold t (omitted if $t=n-1$) and number of parties n, which defines the set of values $\mathcal{M}_{\mathsf{pp}}$ and the predicate classes $\Phi_{\mathsf{pp,iss}}$ and $\Phi_{\mathsf{pp,share}}$ for the predicates accepted during issuance and sharing.

$(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{SSA.KeyGen}(\mathsf{pp})$**:** The keys are generated by *the intermediary sites*.

$(\perp,\sigma) \leftarrow\!\!\$\ \langle \mathsf{SSA.Iss}(\mathsf{pp},\mathsf{sk},\phi) \rightleftharpoons \mathsf{SSA.U}(\mathsf{pp},\mathsf{pk},v,\phi)\rangle$**:** The protocol between *the intermediary sites (as issuers)* and *the users* generates a credential σ[2] bound to $v \in \mathcal{M}$ and $\mathsf{info} \in \{0,1\}^*$. Formally, the protocol execution is r-round and of the following format:

$$(\mathsf{umsg}_1,\mathsf{st}^u) \leftarrow\!\!\$\ \mathsf{SSA.U}_1(\mathsf{pp},\mathsf{pk},v,\phi)\,, \qquad (\mathsf{imsg}_1,\mathsf{st}^i) \leftarrow\!\!\$\ \mathsf{SSA.Iss}_1(\mathsf{pp},\mathsf{sk},\phi,\mathsf{umsg}_1)$$
$$\text{For } i \in [2,r]\text{: } (\mathsf{umsg}_i,\mathsf{st}^u) \leftarrow\!\!\$\ \mathsf{SSA.U}_i(\mathsf{st}^u,\mathsf{imsg}_{i-1}), \qquad (\mathsf{imsg}_i,\mathsf{st}^i) \leftarrow\!\!\$\ \mathsf{SSA.Iss}_i(\mathsf{st}^i,\mathsf{umsg}_i)$$
$$\sigma \leftarrow\!\!\$\ \mathsf{SSA.U}_{r+1}(\mathsf{st}^u,\mathsf{imsg}_r)$$

$(\tau,(\mathsf{ss}_i,\tau_i)_{i\in[n]}) \leftarrow\!\!\$\ \mathsf{SSA.Share}(\mathsf{pp},\mathsf{pk},v,\sigma,\phi)$**:** The *user* generates secret shares $(\mathsf{ss}_i)_{i\in[n]}$ of v, public verification information τ and local verification information τ_i. The public τ can be verified by the *conversion sites*. In contrast, (ss_i,τ_i) are encrypted and only sent to the *aggregation servers* running the MPC protocol.

$0/1 \leftarrow \mathsf{SSA.Ver}(\mathsf{pp},\mathsf{pk},\tau,\mathsf{ss}_i,\tau_i,\phi)$**:** The verification algorithm verifies τ and τ_i for the share ss_i and ϕ.

[2]We use 'credential' and 'signature' interchangeably since the credentials in our constructions are signature over the secret shares.

Game $\mathrm{UNF}^{\mathcal{A}}_{\mathsf{SSA},t,n}(\lambda)$:	Oracle $\mathrm{Iss}_1(\mathrm{sid},\phi,\mathsf{umsg}_1)$:	Oracle $\mathrm{Iss}_j(\mathsf{umsg}_j)$: // $j \in [2,r]$
$Q \leftarrow \varnothing$; $\mathcal{I}_1,\ldots,\mathcal{I}_r \leftarrow \varnothing$	**if** $\mathrm{sid} \in \mathcal{I}_1$ **then abort**	**if** $\mathrm{sid} \in \mathcal{I}_j \ \vee \ \mathrm{sid} \notin \mathcal{I}_1,\ldots,\mathcal{I}_{j-1}$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SSA.Setup}(1^\lambda,t,n)$; $(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{SSA.KeyGen}(\mathsf{pp})$	$Q \leftarrow Q \cup \{\phi\}$	**then abort**
$(\phi^*,\tau^*,(\mathsf{ss}_i^*,\tau_i^*)_{i\in[n]}) \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{Iss}_1,\ldots,\mathrm{Iss}_r}(\mathsf{pp},\mathsf{pk})$	$\mathcal{I}_1 \leftarrow \mathcal{I}_j \cup \{\mathrm{sid}\}$	$\mathcal{I}_j \leftarrow \mathcal{I}_j \cup \{\mathrm{sid}\}$
$v^* \leftarrow \mathsf{SSA.Recover}(\mathsf{pp},(\mathsf{ss}_i^*)_{i\in[n]})$	$(\mathsf{imsg}_1,\mathsf{st}^i_{\mathrm{sid},1}) \leftarrow\!\!\$$	$(\mathsf{imsg}_j,\mathsf{st}^i_{\mathrm{sid},j}) \leftarrow\!\!\$$ // $\mathsf{st}^i_{\mathrm{sid},r} = \perp$
return $(\phi^*(v^*)=0 \ \vee \ \forall \phi \in Q : \phi(v^*)=0) \ \wedge$	$\mathsf{SSA.Iss}_1(\mathsf{pp},\mathsf{sk},\phi,\mathsf{umsg})$	$\mathsf{SSA.Iss}_j(\mathsf{st}^i_{\mathrm{sid},j-1},\mathsf{umsg}_j)$
$\wedge \ \forall i \in [n] : \mathsf{SSA.Ver}(\mathsf{pp},\mathsf{pk},\tau^*,\mathsf{ss}_i^*,\tau_i^*,\phi^*)=1$	**return** imsg_1	**return** imsg_j

Figure 5: Unforgeability game for SSA

$v \leftarrow \mathsf{SSA.Recover}(\mathsf{pp},(\mathsf{ss}_i)_{i\in S})$**:** The recovery algorithm on input shares $(\mathsf{ss}_i)_{i\in S}$ for (an implicit input) $S \subseteq [n], |S| > t$ recovers the value v.

Remark 1 (Verification model)**.** The algorithm SSA.Ver in our constructions will be described as VerPub and VerShare, such that SSA.Ver returns 1 iff both VerPub and VerShare return 1. These algorithms respectively indicate: (1) public verification of τ and predicate ϕ done by the *conversion sites*, and (2) local lightweight verification on each share ss_i using τ,τ_i done by each *aggregation server*.

The two algorithms has the following syntax:

$0/1 \leftarrow \mathsf{SSA.VerPub}(\mathsf{pp},\mathsf{pk},\tau,\phi)$. This algorithm is independent of the shares ss_i and τ_i.

$0/1 \leftarrow \mathsf{SSA.VerShare}(\mathsf{pp},\mathsf{pk},\tau,\mathsf{ss}_i,\tau_i)$.

CORRECTNESS. For any $\lambda,t=t(\lambda)<n=n(\lambda)\in\mathbb{N}$, $\mathsf{pp}\in[\mathsf{SSA.Setup}(1^\lambda,t,n)]$, $v\in\mathcal{M}_{\mathsf{pp}}, S\subseteq[n],|S|>t$, and $\phi\in\Phi_{\mathsf{pp,iss}},\phi'\in\Phi_{\mathsf{pp,share}}$ such that $\phi(v)=\phi'(v)=1$, the following experiment always returns 1.

$$
\begin{aligned}
&(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{SSA.KeyGen}(\mathsf{pp})\\
&(\perp,\sigma) \leftarrow\!\!\$\ \langle \mathsf{SSA.Iss}(\mathsf{pp},\mathsf{sk},\phi) \rightleftharpoons \mathsf{SSA.U}(\mathsf{pp},\mathsf{pk},v,\phi)\rangle\\
&(\tau,(\mathsf{ss}_i,\tau_i)_{i\in[n]}) \leftarrow\!\!\$\ \mathsf{SSA.Share}(\mathsf{pp},\mathsf{pk},v,\sigma,\phi')\\
&\textbf{return } v=\mathsf{SSA.Recover}(\mathsf{pp},(\mathsf{ss}_i)_{i\in S}) \ \wedge \ (\forall i\in S:\mathsf{SSA.Ver}(\mathsf{pp},\mathsf{pk},\tau,\mathsf{ss}_i,\tau_i,\phi')=1)\ .
\end{aligned}
$$

4.2 Unforgeability

At a high-level, the unforgeability property prevents any adversary, who can request credential via the issuance protocol, from producing a the tuple $(\tau^*,(\mathsf{ss}_i^*,\tau_i^*)_{i\in[n]})$ such that (1) each pair (τ^*,τ_i^*) is valid with respect to the shares ss_i^*, and (2) no credential has been issued for the value v^* recovered from ss_i^* (in the more general blind issuance case which we consider, the recovered v^* should not satisfy by prior issuance predicates). The unforgeability game is defined in Figure 5 and the advantage of the adversary $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SSA},t,n}(\mathcal{A},\lambda) := \mathsf{Pr}[\mathrm{UNF}^{\mathcal{A}}_{\mathsf{SSA},t,n}(\lambda)=1]\ .$$

4.3 Unlinkability

At a high-level, unlinkability ensures that no adversary can link the shares (as long as not all of them are revealed) output by the users with a particular issuance session and/or with other shares. The game UNLK is defined in Figure 6 with the adversary's advantage

$$\mathsf{Adv}^{\mathsf{unlk}}_{\mathsf{SSA},t,n}(\mathcal{A},\lambda) := |\mathsf{Pr}[\mathrm{UNLK}^{\mathcal{A}}_{\mathsf{SSA},t,n,0}(\lambda)=1]-\mathsf{Pr}[\mathrm{UNLK}^{\mathcal{A}}_{\mathsf{SSA},t,n,1}(\lambda)=1]|$$

We now explain the game in more details.

Bits β,b_β: The session bit β indicates left (0) and right (1) sessions, while the bit b_β indicates the value v_{b_β} and credential σ_{b_β} that will be used in session β of oracle Share (explained below). The goal of the adversary is to attempt to link the shares back to the issuance sessions which generated them.

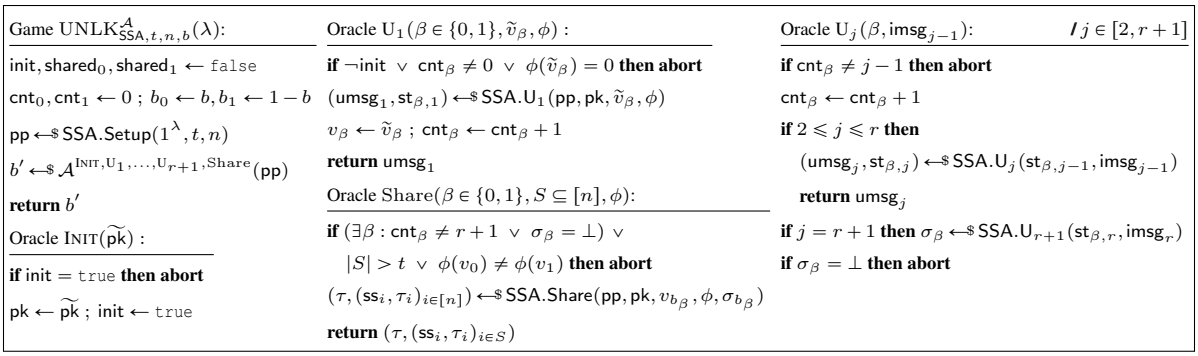


Figure 6: Unlinkability game for SSA.

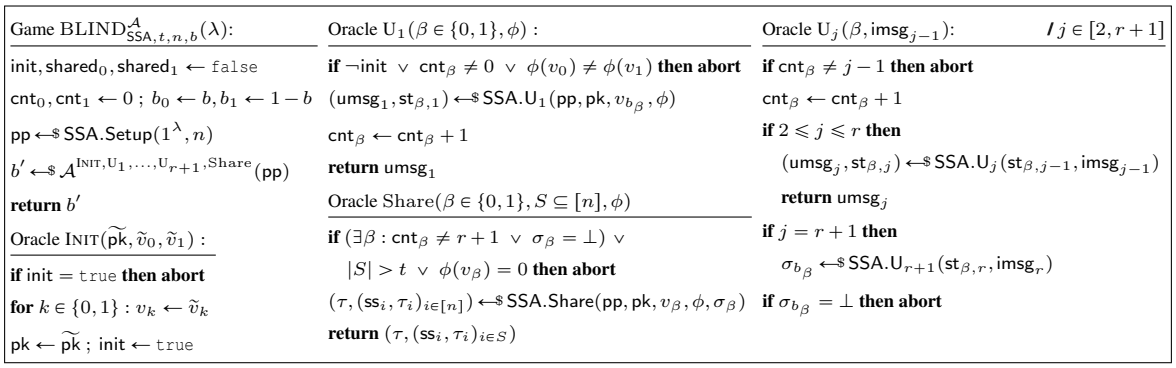


Figure 7: Blind Issuance game for SSA.

Oracle INIT: Pick a malicious public key.

Oracles $\mathrm{U}_1, \ldots, \mathrm{U}_{r+1}$: Initiate an issuance protocol with the users $\beta = 0, 1$ (adversaries can arbitrarily interleave the interactions). The adversary picks the value v_β and the predicate ϕ such that $\phi(v_\beta) = 1$. The resulting credentials are stored by the game as σ_β.

Oracle Share: Request freshly sampled shares $(\mathsf{ss}_i)_{i\in S}$ for a subset S of $[n]$ with $|S| \leqslant t$–this condition incorporates the privacy property of the underlying (threshold) secret sharing scheme–along with the verification information $\tau, (\tau_i)_{i\in S}$. These are computed from the credential σ_{b_β} via the Share algorithm. Note that the bit b_β is initialized depending on which game $\mathrm{UNLK}^{\mathcal{A}}_{\mathsf{SSA},n,b}$ for $b = 0, 1$ the adversary is playing.

The goal of the adversary is to distinguish its interaction with two honest users and whether the produced shares comes from the first or the second user, i.e., correctly guessing the bit b which determines the game.

4.4 Blind Issuance Definition

In addition to unlinkability, we also define the *blind issuance* property, which ensures that the issuance sessions leak no information about the value v *except for what can be inferred by the issuance predicate*. This property has a similar flavor to what is defined for anonymous credentials (e.g., in [CKL+16]). We remark that our constructions fully reveal the value v at issuance through the predicate; and hence, blind issuance is trivially satisfied.The games BLIND is described in Figure 7 with the adversary's advantage defined as

$$\mathsf{Adv}^{\mathrm{blind}}_{\mathsf{SSA},t,n}(\mathcal{A}, \lambda) := |\mathsf{Pr}[\mathrm{BLIND}^{\mathcal{A}}_{\mathsf{SSA},t,n,0}(\lambda) = 1] - \mathsf{Pr}[\mathrm{BLIND}^{\mathcal{A}}_{\mathsf{SSA},t,n,1}(\lambda) = 1]| \,.$$

Similarly to unlinkability, we also consider two notions, blind issuance and one-time show blind issuance. The distinction from the unlinkability game is that now, the adversary is trying to guess which value v_0, v_1 (of its choice) is used by which user $\beta = 0$ or $\beta = 1$.

We now explain the game, the oracle access, and the distinctions between blind isssuance and unlinkability:

Oracle INIT: The adversary picks a malicious public key and also two values v_0, v_1.

Bits β, b_β: The bit β has different meanings with regard to oracle U and Share. In particular, for U, β denotes left and right sessions where the adverary "does not" know which value v_0, v_1 is being used–this being determined by the bit b_β. In contrast, β in Share indicates which value v_β is being used. The bit b_β is set to β or $1-\beta$ depending on which game BLIND_0 or BLIND_1 the adversary is playing.

Oracles $\text{U}_1, \ldots, \text{U}_{r+1}$: Initiate an issuance protocol with the users $\beta = 0, 1$ (adversaries can arbitrarily interleave the interactions). The adversary picks the predicate ϕ such that $\phi(v_0) = \phi(v_1)$, i.e., imposing the condition that the issuance predicate should not leak the value being used. The game uses v_{b_β} to run the user-side algorithms and store the resulting credential as σ_{b_β}. Note that this is different from the unlinkability game where the adversary "knows" that the value v_β is being used for session β.

Oracle Share: Request freshly sampled shares $(\mathsf{ss}_i)_{i \in S}$ for a subset S of $[n]$ with $|S| \leqslant t$ along with the verification information $\tau, (\tau_i)_{i \in S}$. These are computed from the credential σ_β and the value v_β via the Share algorithm. The predicate ϕ specified here need not be satisfied by both v_0 or v_1. We emphasize that the adversary does know that v_β is being used; however, it's goal is to determine whether σ_β is issued in the left (0) or right (1) issuance sessions. This is in contrast to the unlinkability where the adversary "does not" know the value (the game uses v_{b_β}) and the predicate is constrained to not reveal the value used.

Note that the blind issuance property is more akin to the blindness definition of blind signatures [Cha82, JLO97].

5 BBS-Based SSA

In Figure 8, we describe the BBS-based scheme $\mathsf{SSA}_{\mathsf{BBS}}$ which allows attesting additive secret shares of values $\vec{v} \in \mathbb{Z}_p^m$ for integer $m \geqslant 1$ (extending from $v \in \mathbb{Z}_p$ as sketched in the overview). The scheme also allows embedding a public tag info (e.g., date and time) as an attribute of the credentials. In particular, the issuance and sharing phases' predicate classes contain predicates of the form $\phi_{\underline{\mathsf{info}},\underline{\vec{v}}}$ and $\phi_{\underline{\mathsf{info}}}$, resp., defined as

$$\phi_{\underline{\mathsf{info}},\underline{\vec{v}}}((\mathsf{info}, \vec{v})) := (\mathsf{info} = \underline{\mathsf{info}}) \ \wedge \ (\vec{v} = \underline{\vec{v}}) \,, \tag{1}$$

$$\phi_{\underline{\mathsf{info}}}((\mathsf{info}, \vec{v})) := (\mathsf{info} = \underline{\mathsf{info}}) \,. \tag{2}$$

To produce verification information τ, τ_i, our construction makes use of the proof system Π_{Share} (described in Appendix A.2 based on Σ-protocols), which proves knowledge of witnesses corresponding to the following relation:

$$\mathsf{R}_{\mathsf{Share}} := \left\{ \begin{array}{l} ((G, \vec{H}, \widetilde{A}, \widetilde{B}, (C_i)_{i \in [n]}, \mathsf{info}); \\ (\alpha, e, \vec{v} \in \mathbb{Z}_p^m, \vec{s} \in (\mathbb{Z}_p^m)^n, \vec{r})) \end{array} : \begin{array}{l} \widetilde{B} = \alpha(G + \sum_{j=1}^m v_j H_j + \mathsf{info} H_{n+1}) - e\widetilde{A} \wedge \\ \vec{v} = \sum_{i=1}^n \vec{s}_i \ \wedge \ \forall i \in [n] : C_i = r_i G + \sum_{j=1}^m s_{i,j} H_j \end{array} \right\}.$$

We refer to Section 2.3 for a detailed overview of our protocol. At a high-level, the issuer first sends a BBS signature of the pair $(\mathsf{info}, \vec{v})$ to the user. To share, the user computes secret shares $\vec{s}_i$ of $\vec{v}$ and Pedersen commitments C_i to each share along with proof of knowledge of valid signature, secret shares, and openings with respect to $\mathsf{R}_{\mathsf{Share}}$ (we take the approach from the PoK in [TZ23]). This is an optimization over naively and locally proving partial-opening for each $i \in [n]$. Hence, our VerShare only checks commitment openings.

Theorem 2. The scheme $\mathsf{SSA}_{\mathsf{BBS}}$ satisfies correctness, unforgeability, unlinkability, and blind issuance.

Proof sketch. The full proof is in Appendix A.3. **Correctness** follows from correctness of BBS signatures and completeness of the proof. **Unforgeability** follows from unforgeability of BBS signatures and straight-line knowledge-soundness Π_{Share}. **One-time unlinkability** follows from zero-knowledge of Π_{Share}, privacy of additive secret sharing, and perfect hiding of Pedersen commitments. **Blind issuance** is trivially satisfied. □

6 Equivalence Class Signatures for Commitments

In this section, we define equivalence class signatures for commitments (SEQ) in Section 6.1 and give a construction of such signature scheme in Section 6.2 which signs Pedersen commitments and allow the signature and message to

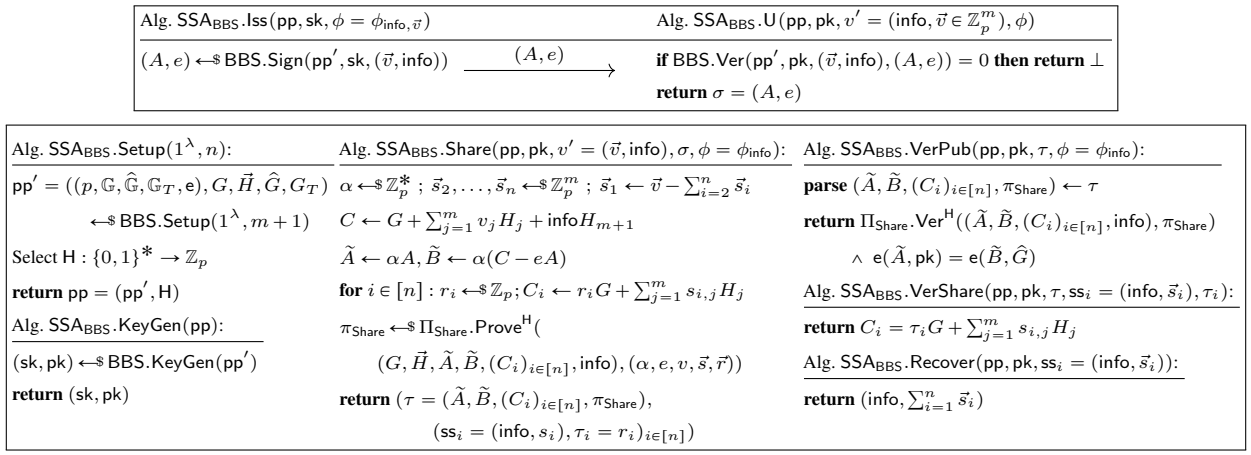


Figure 8: Description of $\mathsf{SSA}_{\mathsf{BBS}}=\mathsf{SSA}_{\mathsf{BBS}}[\mathsf{GGen},n,m]$ generating n-out-of-n additive secret shares for $\vec{v}\in\mathbb{Z}_p^m$ using the signature scheme $\mathsf{BBS}=\mathsf{BBS}[\mathsf{GGen},m+1]$. Note: the user sends $\vec{v}$ and info as its first move.

be adapted within some defined equivalence class. Note that the syntax and definition are inspired from Signatures on Rerandmizable Ciphertexts (SoRC) [BF20]. However, our scheme is defined with respect to a rerandomizable commitment scheme, and our (adapted) signatures can be verified with the (rerandomized) commitments.

6.1 Definition

SYNTAX. The primitive SEQ has the following syntax:

$\mathsf{pp}\leftarrow\$\ \mathsf{SEQ.Setup}(1^\lambda,n)$**:** The public parameters implicitly define the message space $\mathcal{M}$, the equivalence classes (denoted $[\![\vec{m}]\!]_{\mathsf{pp}}$ for messages $\vec{\mu}\in\mathcal{M}^n$), the randomness space for the commitment and the adaption $\mathcal{R}^{\mathsf{com}},\mathcal{R}^{\mathsf{Adapt}}$, and the public key space $\mathcal{K}$.

$C\leftarrow\mathsf{SEQ.Com}(\mathsf{pp},m\in\mathcal{M};r\in\mathcal{R}^{\mathsf{com}})$**:** The commit algorithm. We also write $(C,r)\leftarrow\$\ \mathsf{SEQ.Com}(\mathsf{pp},m)$ specifying that the algorithm samples the randomness r. We may write $(\vec{C},\vec{r})\leftarrow\$\ \mathsf{SEQ.Com}(\mathsf{pp},\vec{\mu}\in\mathcal{M}^n)$ or $\vec{C}\leftarrow\mathsf{SEQ.Com}(\mathsf{pp},\vec{\mu},\vec{r})$ for committing to the vector of messages $\vec{m}$ element-wise.

$(\mathsf{sk},\mathsf{pk})\leftarrow\$\ \mathsf{SEQ.KeyGen}(\mathsf{pp})$**:** The key generation. The membership $\mathsf{pk}\in\mathcal{K}$ can be efficiently checked.

$\sigma\leftarrow\$\ \mathsf{SEQ.Sign}(\mathsf{pp},\mathsf{sk},\vec{C})$**:** The signing algorithm.

$0/1\leftarrow\mathsf{SEQ.Ver}(\mathsf{pp},\mathsf{pk},\vec{C},\sigma)$**:** Verify that the signature σ is valid for the commitment $\vec{C}$.

$(\vec{\mu}',\vec{r}',\vec{C}',\sigma')\leftarrow\$\ \mathsf{Adapt}(\mathsf{pp},\mathsf{pk},\vec{C},\vec{\mu},\vec{r},\sigma,\mathsf{rnd}\in\mathcal{R}^{\mathsf{Adapt}})$**:** Rerandomize the signature along with the message $\vec{\mu}$, the randomness $\vec{r}$, and the commitment $\vec{C}$.

$0/1\leftarrow\mathsf{SEQ.VerAdapt}(\mathsf{pp},\mathsf{pk},\vec{C}',\sigma')$**:** Verify that the adapted signature σ is valid for the commitment $\vec{C}'$.

Note that it is not necessary that σ' from Adapt can be further adapted. In fact, σ' will be shorter than σ in our construction. We also define our unforgeability with respect to VerAdapt (which is a stronger guarantee).

CORRECTNESS. For any $\lambda,n=n(\lambda)\in\mathbb{N},\mathsf{pp}\in[\mathsf{SEQ.Setup}(1^\lambda,n)],\vec{\mu}\in\mathcal{M}^n$, the following experiment always returns 1.

$$\begin{aligned}&(\mathsf{sk},\mathsf{pk})\leftarrow\$\ \mathsf{SEQ.KeyGen}(\mathsf{pp})\\&(\vec{C},\vec{r})\leftarrow\$\ \mathsf{SEQ.Com}(\mathsf{pp},\vec{\mu});\sigma\leftarrow\$\ \mathsf{SEQ.Sign}(\mathsf{pp},\mathsf{sk},\vec{C})\\&\textbf{return}\ \mathsf{SEQ.Ver}(\mathsf{pp},\mathsf{pk},\vec{C},\sigma)\ .\end{aligned}$$

ADAPTION. For all $\mathsf{pp}\in[\mathsf{SEQ.Setup}(1^\lambda,n)]$, $\mathsf{pk}\in\mathcal{K}$, $\vec{\mu}\in\mathcal{M}^n$, $\vec{r}\in(\mathcal{R}^{\mathsf{com}})^n,\vec{C}=\mathsf{Com}(\mathsf{pp},\vec{\mu};\vec{r})$, and σ such that $\mathsf{SEQ.Ver}(\mathsf{pp},\mathsf{pk},\sigma,\vec{C})=1$, the following distributions are indisguishable:

$$\mathcal{D}_0:=\left\{(\vec{\mu}',\vec{r}',\vec{C}',\sigma'):\ \begin{array}{l}\vec{\mu}'\leftarrow\$\ [\![\vec{\mu}]\!]_{\mathsf{pp}}\ ,(\vec{C}',\vec{r}')\leftarrow\$\ \mathsf{SEQ.Com}(\mathsf{pp},\vec{\mu}')\\ \sigma'\leftarrow\$\ \{\sigma':\mathsf{SEQ.VerAdapt}(\mathsf{pp},\mathsf{pk},\vec{C}',\sigma')=1\}\end{array}\right\},$$

Game $\mathrm{UNF}^{\mathcal{A}}_{\mathsf{SEQ},n}(\lambda)$:	Oracle $\mathrm{S}(\vec{\mu},\vec{r})$:
$Q \leftarrow \emptyset$; $\mathsf{pp} \leftarrow\!\!\$\, \mathsf{SEQ.Setup}(1^\lambda, n)$	$C = \mathsf{SEQ.Com}(\mathsf{pp}, \vec{\mu}; \vec{r})$
$(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\, \mathsf{KeyGen}(\mathsf{pp})$	$Q \leftarrow Q \cup \{\vec{\mu}\}$
$(\vec{\mu}^*, \vec{r}^*, \sigma^*) \leftarrow\!\!\$\, \mathcal{A}^{\mathrm{Iss}}(\mathsf{pp},\mathsf{pk})$	$\sigma \leftarrow\!\!\$\, \mathsf{SEQ.Sign}(\mathsf{pp},\mathsf{sk},C)$
$\vec{C}^* \leftarrow \mathsf{SEQ.Com}(\mathsf{pp}, \vec{\mu}^*; \vec{r}^*)$	**return** σ
return $(\forall \vec{\mu} \in Q : \vec{\mu}^* \notin [\![\vec{\mu}]\!]_{\mathsf{pp}})$	
$\wedge\ \mathsf{SEQ.Ver}(\mathsf{pp},\mathsf{pk},\vec{C}^*,\sigma^*) = 1$	

Figure 9: Unforgeability game of SEQ.

$$\mathcal{D}_1 := \left\{ (\vec{\mu}', \vec{r}', \vec{C}', \sigma') : \mathsf{rnd} \leftarrow\!\!\$\, \mathcal{R}^{\mathsf{Adapt}}, (\vec{\mu}', \vec{r}', \vec{C}', \sigma') \leftarrow\!\!\$\, \mathsf{Adapt}(\mathsf{pp},\mathsf{pk},\vec{C},\vec{\mu},\vec{r},\sigma,\mathsf{rnd}) \right\} .$$

We denote the distinguishing advantage of any adversary $\mathcal{A}$ as $\mathsf{Adv}^{\mathsf{adapt}}_{\mathsf{SEQ}}(\mathcal{A},\lambda)$. A SEQ scheme satisfies perfect adaption if the two distributions are identical. Essentially, adaption says that the adapted signatures are distributed similarly to a "fresh" signature. In this case, fresh means a signature that is randomly sampled from the space of valid adapted signatures. Note that this covers any maliciously generated public key $\mathsf{pk} \in \mathcal{K}$ and signatures σ valid with respect to pk.

UNFORGEABILITY. Unforgeability game is defined in Figure 9. This says that no adversary, who has signing query access where the queries require opening the commitments to be signed, cannot produce a valid forgery (with respect to SEQ.VerAdapt not SEQ.Ver) on a vector of message not in the equivalence classes of any of the signed messages. The advantage is

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SEQ},n}(\mathcal{A},\lambda) := \Pr[\mathrm{UNF}^{\mathcal{A}}_{\mathsf{SEQ},n}(\lambda) = 1] .$$

HIDING OF COMMITMENT. Hiding for commitments is defined as usual in that the commitment does not reveal any information about the message. For all $\lambda, n = n(\lambda) \in \mathbb{N}, \mathsf{pp} \in [\mathsf{SEQ.Setup}(1^\lambda, n)]$ and $m, m' \in \mathcal{M}$, the following distributions are indistinguishable

$$\{C : (C,r) \leftarrow\!\!\$\, \mathsf{SEQ.Com}(\mathsf{pp}, m)\} , \qquad \{C' : (C',r') \leftarrow\!\!\$\, \mathsf{SEQ.Com}(\mathsf{pp}, m')\}$$

We denote the distinguishing advantage of any adversary $\mathcal{A}$ as $\mathsf{Adv}^{\mathrm{hid}}_{\mathsf{SEQ}}(\mathcal{A},\lambda)$. We say that the commitment is perfectly hiding if the distributions are identical.

6.2 Construction

In Figure 10, we describe the $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, m, n, A]$ construction which signs Pedersen commitments $\vec{C}$ of messages $\vec{\mu} \in (\mathbb{Z}_p^m)^n$ with equivalence classes defined by a full-row-rank matrix $A \in \mathbb{Z}_p^{\ell \times n}$.[3] The matrix A will be dependent on the application, but generally it defines the corresponding equivalence class $[\![\vec{\mu}]\!]_{\mathsf{pp}}$ of $\vec{\mu} = (\vec{\mu}_i)_{i\in[n]}$ as

$$\left\{ \left(\vec{\mu}_i + \textstyle\sum_{j=1}^{\ell} A_{j,i} \vec{\alpha}_j \right)_{i\in[n]} : (\vec{\alpha}_i)_{i\in[\ell]} \in (\mathbb{Z}_p^m)^\ell \right\} . \tag{3}$$

To give an example, when $m = 1$, we have

$$[\![\vec{\mu}]\!]_{\mathsf{pp}} := \left\{ \vec{\mu} + A^T \vec{\alpha} : \vec{\alpha} \in \mathbb{Z}_p^\ell \right\}, \tag{4}$$

containing vectors shifted from $\vec{\mu}$ by the images of A^T. The construction relies on similar ideas from SoRC [BF20] as sketched in Section 2. Our main contribution is that achieving a signature scheme for a larger family of equivalence classes based on linear transformation over $(\mathbb{Z}_p^m)^n$. The following theorem, proved in Appendix B, establishes our security with unforgeability proved in the algebraic group model (AGM) [FKL18].

Theorem 3. For a full-row-rank matrix $A \in \mathbb{Z}_p^{\ell\times n}$, the construction $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, m, n, A]$ in Figure 10 satisfies correctness, perfect adaption, perfect hiding, and unforgeability (in the AGM) from $(3q, q+1)$-DL assumption.

[3] The message space is $\mathcal{M} = \mathbb{Z}_p^m$ and equivalence classes are defined over vectors of $\mathcal{M}^n$. Also, the randomness space is $\mathbb{Z}_p$, and the randomness space for adaption is $(\mathbb{Z}_p^m)^\ell \times \mathbb{Z}_p^n$.

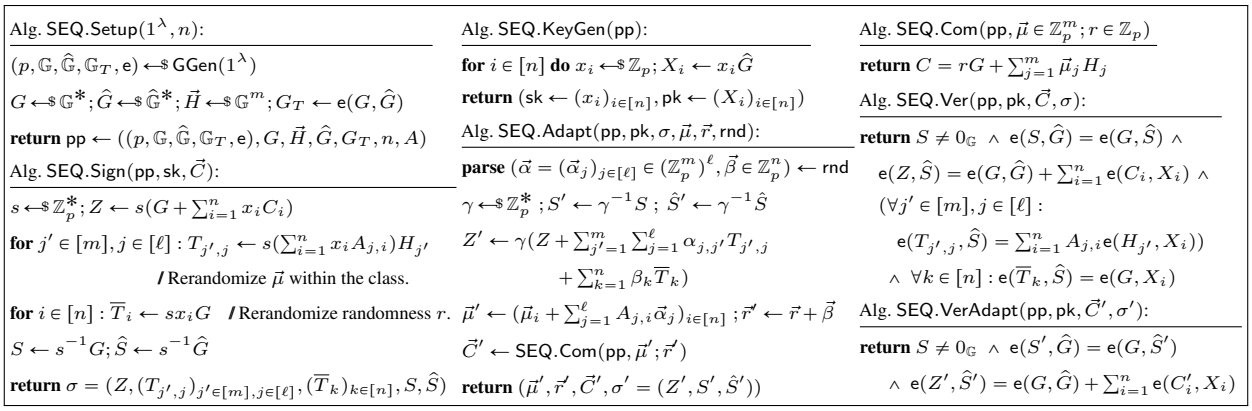
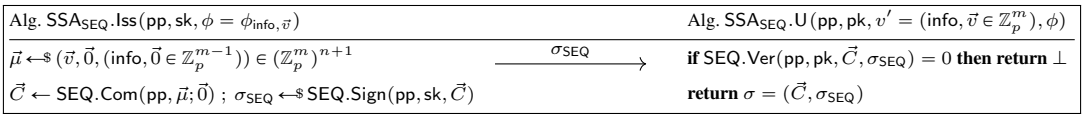


Figure 10: Construction of $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, m, n, A]$ for signature for additive equivalence class over $(\mathbb{Z}_p^m)^n$ messages defined by integers $n = n(\lambda), m = m(\lambda)$, and the matrix $A \in \mathbb{Z}_p^{\ell \times n}$ for $\ell < n$.



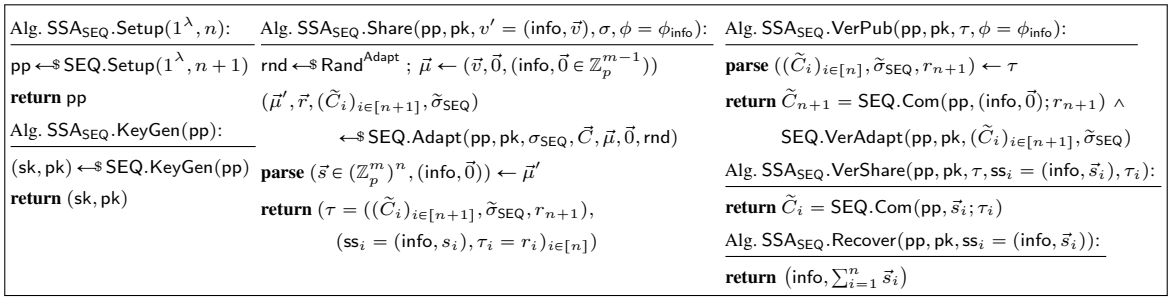


Figure 11: Description of $\mathsf{SSA}_\mathsf{SEQ} = \mathsf{SSA}_\mathsf{SEQ}[\mathsf{GGen}, m, n]$ for generating and authenticating n-of-n secret shares. Denote $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, m, n+1, A]$ with $A = [\vec{1} \| -\mathbb{I}_{n-1} \| \vec{0}] \in \mathbb{Z}_p^{(n-1) \times (n+1)}$. Note: the user only sends $\vec{v}$ and info in the first move.

7 SEQ-Based SSA

In this section, we describe in Figure 11 our SEQ-based SSA construction $\mathsf{SSA}_\mathsf{SEQ}$. This construction does not rely on proofs of knowledge, and the verification only performs algebraic computations (i.e., multi-exponentiations and pairing evaluations). Similar to the scheme in Section 5, the predicate classes for issuance and sharing are the ones in Equations (1) and (2), allowing a public tag info to be bound to the credential along with $\vec{v} \in \mathbb{Z}_p^m$. As a result, each equivalence class (for SEQ) is defined via the pair $(\mathsf{info}, \vec{v})$ such that

$$[\![(\mathsf{info}, \vec{v})]\!]_{n,m} := \left\{(\vec{s} \in (\mathbb{Z}_p^m)^n, \mathsf{info}) : \sum_{i=1}^n \vec{s}_i = \vec{v}\right\} .$$

In the sense of Equation (3), each class is constrained by the linear map $A = [\vec{1} \| -\mathbb{I}_{n-1} \| \vec{0}] \in \mathbb{Z}_p^{(n-1) \times (n+1)}$. The last column of A corresponds to no rerandomization of info.

To produce shares, the user randomly adapts the messages, commitments and signatures, resulting in the shares $\vec{s}_i$, the randomness r_i for the commitments $\widetilde{C}_i$ and the rerandomized signature $\widetilde{\sigma}_\mathsf{SEQ}$. These are then returned as the verification information τ and τ_i's along with the shares $\vec{s}_i$. Verification simply verifies the signatures and for each $i \in [n]$ checks that the commitment openings are valid.

Theorem 4. The scheme $\mathsf{SSA}_\mathsf{SEQ}$ in Figure 11 satisfies correctness, unforgeability, unlinkability and blind issuance.

Proof sketch. The full proof is in Appendix C. **Correctness** follows from correctness and adaption of SEQ.

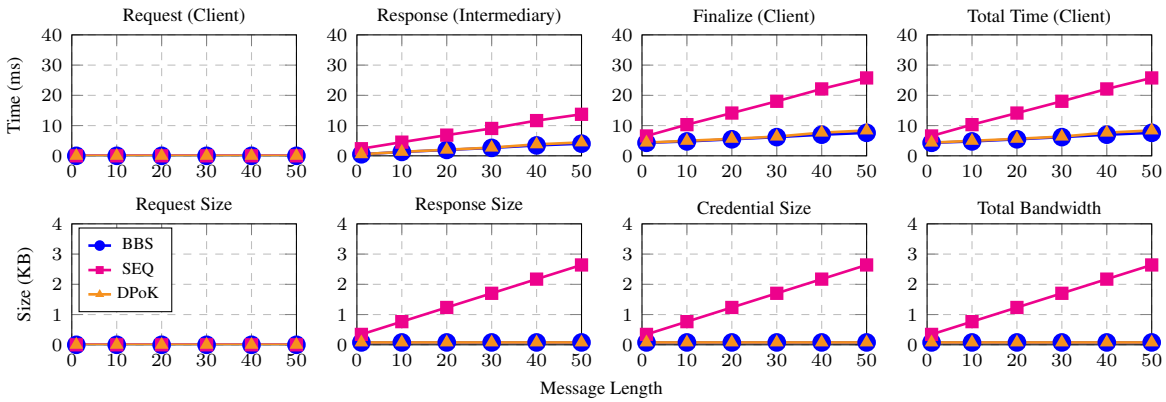


Figure 12: Issuance costs for the client and issuer to acquire credentials over a single message of varying length. Bandwidth costs do not include the cost to send the message itself.

Unforgeability follows from unforgeability of SEQ. In particular, we construct a reduction such that (1) the issuance is simulated via the signing oracle on the vector $(\vec{v}, \vec{0}, (\mathsf{info}, \vec{0}))$ and randomness $\vec{0}$, (2) from a forgery $(\tau = ((\widetilde{C}_i)_{i\in[n+1]}, \sigma, r_{n+1}), (\vec{s}_i, \tau_i = r_i)_{i\in[n]})$ for predicate $\phi^* = \phi_{\mathsf{info}}$, it extracts a SEQ forgery. This is because if the forgery is valid and fresh, the recovered values info or $\vec{v} = \sum_{i=1}^{n} \vec{s}_i$ were not queried before.

Unlinkability follows from privacy of the additive secret sharing scheme, adaption of SEQ, and hiding commitment of SEQ. More precisely, adaption ensures that the adapted signature remains indistinguishable to a freshly generated one, hiding ensures that the unopened commitment reveals nothing about the hidden shares, and privacy of the additive secret sharing ensures that the values $\vec{v}_0, \vec{v}_1$ remains hidden. **Blind issuance** is trivially satisfied. □

8 Evaluation

We evaluate our proposed constructions in the context of a common mode of use of PPA in computing histograms. The measurement takes the form of a vector where each index corresponds to an attribution target (e.g., an ad campaign) and the value in the index represents how much value to attribute to the target. The aggregation service simply sums the measurements and releases a histogram to the conversion site. In an attestation solution, intermediary sites would sign one-hot vectors (vectors in which one index is set to one while the rest are zero) corresponding to the index of the served ad. Users would send attested shares of one-hot vectors along with public values to scale each vector. In an unattested solution, users would submit secret shares of a vector and prove that the L1 norm is bounded by a public value. Alternatively, users would submit secret shares of one-hot vectors along with public values to scale each vector and prove that the L1 norm is bounded by one. We consider the *two-server MPC aggregation model* as that is the model focused on in the standards process [BCPS25].

Our evaluation aims to answer the following questions:

- What are the computation and bandwidth cost profiles of our attestation constructions across different parties in the PPA ecosystem (user client, issuer, conversion server, aggregation servers).
- How does the cost scale with histogram size?
- How does the cost profile compare with that of alternative attestation solutions?
- How does the cost profile compare with that of alternative unattested solutions?

8.1 Implementation

We implement our BBS SSA protocol (Figure 8) and our SEQ SSA protocol (Figure 11) in Rust using the `arkworks` library for pairing-friendly curves [ark]. The zero-knowledge proofs for the BBS solution are instantiated using standard knowledge of discrete log representation Σ-protocols and made non-interactive using the Fiat-Shamir transform.

We implement the CDL approach for BBS opening [CDL16] (adapted from BBS+ [TZ23]) which is the current approach taken in the ongoing IETF standard on verifiable credentials [LKWL24]. We implement a standard pairing verification optimization for the SEQ scheme to batch pairing checks together using a random linear combination. The implementations are generic to the choice of pairing-friendly curve (and hash function for Fiat-Shamir); we report on an instantiation over BLS12-381 and Blake2 to target 128 bits of security. Benchmarks are performed single-threaded on a 2.4GHz Intel Core i5 processor with 16GB of RAM [4] .

BASELINES. We compare our constructions against alternate attested and unattested solutions. For unattested comparisons, we evaluate the Prio3 system with input validity proofs for L1 norm bounding. The proof system used in the standards draft is derived from the fully linear proofs of Boneh et al. [BBC+19, BCPS25, TC25]; our evaluation uses the reference implementation [lib]. We consider L1 bounds of 1-bit (i.e., one-hot) and of 8-bits.

For comparison against an attested solution, we implement a variant of the distributed proof of knowledge (DPoK) of BBS signature construction proposed by Dutta et al. [DGPS24]. In this solution, users receive a BBS signature on the plaintext message. To open the message, the user provides shares to each aggregation server along with a blinded signature. The servers run an interactive DPoK protocol to the verify the blinded signature using their shares. We distribute the CDL BBS opening proof [CDL16, TZ23] extending techniques from the FROST distributed Schnorr signature protocol [KG20, CKK+25]. Dutta et al. propose further optimizations to proof size by compressing the proof size from linear in the opening length to logarithmic using compressed Σ-protocols [AC20]; we do not implement this optimization, and we note that this optimization would also apply to the Σ-protocols used in our BBS SSA solution.

While our evaluation does not evaluate the cost of sending the shares themselves, we note that distributed point functions (DPFs) are a common optimization for compressing shares of one-hot vectors [GI14] which can be efficiently verified [dCP22, DPRS23, MST24]. DPFs can also be used to compress attested one-hot secret shares where the server expands the DPF before verifying attestation. We do not implement DPF compression.

8.2 Issuance Costs

The BBS SSA solution and DPoK solution implement the same issuance procedure in which the intermediary simply sends the client a constant-size BBS credential on the message. Computation for creating and verifying this BBS credential grows linearly in the message length but is concretely inexpensive. Figure 12 shows < 10ms for a message of length 50. Unattested solutions do not have an issuance phase and are not represented in Figure 12.

In contrast, the SEQ SSA solution incurs linear (in the message length) issuance bandwidth since the SEQ signature includes a linear number of "adaption" group elements. Computation costs to create and verify these elements are also somewhat more expensive than the BBS-based solutions: $\approx 3.4\times$ for both the intermediary and client. Figure 12 shows that for a message of length 50, client computation is under 30ms and total bandwidth is under 4KB.

We also note that none of the schemes require a client request for issuance for this application, since the issued message is fully known to the intermediary and no blind signature proofs of knowledge are needed; issuance can be initiated by the issuer without client interaction. Some extensions discussed in Section 9 such as supporting rate limiting would incur costs during request.

8.3 Opening Costs

The BBS SSA solution and the DPoK solution both produce a BBS signature proof of opening of size linear in the message length. The key difference is that the BBS SSA solution has the client produce the proof such that it is verified by the conversion server, while the DPoK solution has the aggregation servers jointly produce the proof in an interactive protocol. Figure 13 shows that the total costs for these two approaches are comparable, incurring bandwidth costs of around 4KB and verification computation costs of less than 40ms for a message of length 50. However, operationally, where this work is incurred is different. We expand on the operational benefits of our non-interactive SSA model in Section 9

[4] Our full implementation is available in an open-source repository (https://github.com/nirvantyagi/ppa-fraud-ssa). It includes instructions for rerunning the benchmarks and reproducing the plots provided.

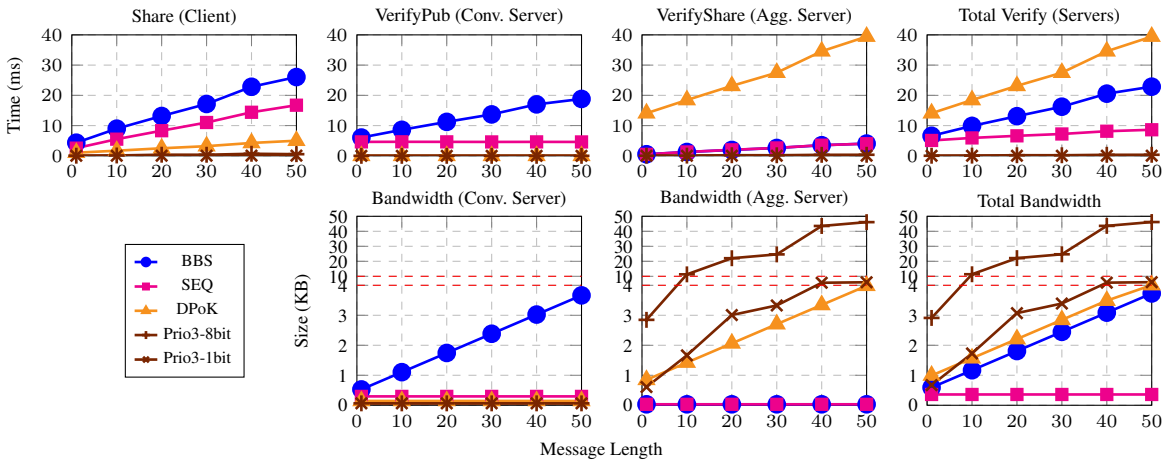


Figure 13: Opening and verification costs for the client, conversion server, and aggregation servers. Bandwidth costs do not include the cost of sending the secret shares themselves.

The SEQ SSA solution is less expensive at verification time, incurring only constant size bandwidth overhead of 296B for opening regardless of message length (not including the size of the shares themselves) and < 10ms verification for a message of length 50.

Figure 13 also compares against the unattested Prio3 solution proving L1 norm bounds for inputs. The computation costs for these solutions are minimal as they only incur very efficient field operations as opposed to the more expensive elliptic curve group operations of the attested solutions. However, they still incur comparable bandwidth costs for aggregation servers (e.g., 6KB and 45KB for 1-bit and 8-bit L1 norm verification of message of length 50). Such proofs provide a different integrity guarantee than attestation. When the attesting issuer is trusted, attestation solutions can discard additional wellformedness proofs.

9 Discussion

In this section, we explore possible extensions and future work, and also discuss further benefits of our solutions.

RATE LIMITING AND REPLAY PREVENTION. Even if an impression is authentic (in that it was actually served to a user), it may not be included in a report in an authentic way. Bad actors may spam report a single impression to artificially boost its conversion value. Standard techniques for rate limiting and expiration can be employed. Rate limiting [CHK+06, YW25] techniques involve adding a PRF key to the credential and providing (and proving) one-time-use nullifiers computed as PRF evaluations of a range-bounded nonce. Expiration can be handled using the public info field [TCR+22, HW25] of our protocols by encoding epoch information for the validity of an impression.

BINDING IMPRESSIONS. Multi-touchpoint user journeys are a useful tool in detecting inauthentic behavior. A multi-touchpoint journey might include an initial impression for clicking the ad, a second impression for spending longer than 5 seconds on the ad destination website, and a third impression for interacting with content on the ad destination website. Authenticating a journey can require a user to provide all three impressions, but these impressions must also be bound to each other to prevent mix-and-match attacks to trivially create journeys. Prior approaches to binding impressions together work by including a shared nonce that is blind to the issuer [IT19]; such an approach could be adapted to our setting, e.g., via blind issuance protocols for BBS signatures [ASM06, CDL16, TZ23] or equivalence class signatures [FHS19].

HIDING ISSUERS AND DUMMY IMPRESSIONS. Our approach relies on messages being attested by third parties that are in turn identified by public keys. If an attribution report is requested from a user but they do not have an attested impression, the lack of an impression can leak information about a user's browsing habits. This could be remediated by obtaining a "dummy" impression from the conversion site (e.g., one that signs over the zero-vector). Even so, our protocol does not hide the issuer. The public key used for verification identifies the intermediary (or dummy-signer) and

can leak browsing information. Future work might adopt ideas from group signatures [CvH91, Gro07, TLMR22] to hide the issuer identity among a set of certified issuers.

DELAYED DISCOVERY OF AUTHENTICITY. Our current approach relies on intermediaries to determine user authenticity and issue impression credentials online. In practice, many fraud detection systems operate offline by running a host of expensive classifiers and aggregating data from different sources. This approach falls outside the capabilities of our proposal and would likely require different techniques to enable, e.g., running a separate aggregation service for this purpose [TTS+23].

PAIRING-FREE CONSTRUCTIONS. Without standardized pairing-friendly curves, constructions from pairing-free elliptic curves may be more desirable. It is easy to adapt our BBS-based SSA to use pairing-free curves in the server-aided model [CHLT25, DDKT25]. However, this approach incurs extra communication with the intermediary sites to produce one-time use material for generating verification information.

OPERATIONAL BENEFITS. A key difference between our SSA solutions and prior work on certified inputs like the DPoK solution of Dutta et al. [DGPS24] is where the burden of verification lies. In DPoK, aggregation servers perform the verification procedure. In practice, aggregation servers may be run by third-party organizations (e.g., the ISRG in Mozilla's deployment of Prio3); minimizing their workload is a key constraint in the system design. In both of our SSA schemes, the bulk of attestation verification is offloaded to the conversion server, which better aligns the protocol workload on the party that benefits from the protocol outputs. The aggregation server need only open a single Pedersen commitment.

Interactive verification via DPoK is comparable to the interactive verification costs of wellformedness proofs over secret shares [BBC+19] (see Figure 13). These costs become significant for large message batches (e.g., of size 10^6-10^9 [Mas25]). The naive proving approach of repeating each interactive verification protocol in parallel would scale bandwidth linearly with message batch size. Alternatively, proof batching can be employed to verify all messages together, but then the rounds of interaction scale logarithmically (in the message batch size) to identify and discard messages with failed proofs via binary search.

Our SSA solutions open up an entirely new deployment scenario that eliminates the need for interaction between aggregation servers. For common affine aggregation functions (like summation) [CB17], the only interaction needed between aggregation servers is for verifying input wellformedness. With SSA, aggregation servers perform lightweight SSA verification and compute their share of the aggregation function non-interactively, sending the result back to the conversion server. This strategy only works when considering security against honest-but-curious aggregation servers; SSA does not provide any guarantee that the aggregation server actually inputs the attested share into the MPC protocol (whereas certified input solutions with interactive verification do provide this guarantee). Nevertheless, the honest-but-curious aggregation server security model is the main focus of the standards effort and we see non-interactive aggregation servers as an attractive deployment model for the PPA ecosystem.

Note, this direction also opens up an efficient batching solution as well. In particular, the conversion server receives multiple attribution reports with the verification information $\tau^{(j)}$ (j indexing each reports), where in our constructions would contain Pedersen commitments $(C_i^{(j)})_{i\in[n]}$. Then, it can verify each $\tau^{(j)}$ with VerPub and for each $i \in [n]$ linearly combine $C_i^{(j)}$ into C'_i, which is sent later to each aggregation server. The aggregation server i then similarly linearly combines their $(\mathsf{ss}_i^{(j)}, r_i^{(j)})$ into (ss'_i, r'_i) and simply checks $C'_i = \mathsf{ss}'_i H + r'_i G$. This saves time on the aggregation server side. However, determining which $\mathsf{ss}_i^{(j)}$ causes the check to fail instead requires additional (logarithmic) rounds of interaction.

Acknowledgements

The authors wish to thank Roxana Geambasu, Martin Thomson, and Charles Harrison for independent discussions regarding the modeling of our approach. Chairattana-Apirom and Tessaro's research was partially supported by NSF grants CNS-2026774, CNS-2154174, CNS-2426905, a gift from Microsoft, and a Stellar Development Foundation Academic Research Award.

References

[AC20] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Cham, August 2020.

[ADEO21] Diego F. Aranha, Anders P. K. Dalskov, Daniel Escudero, and Claudio Orlandi. Improved threshold signatures, proactive secret sharing, and input certification from LSS isomorphisms. In Patrick Longa and Carla Ràfols, editors, *LATINCRYPT 2021*, volume 12912 of *LNCS*, pages 382–404. Springer, Cham, October 2021.

[ark] `arkworks` zksnark ecosystem. https://arkworks.rs.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 111–125. Springer, Berlin, Heidelberg, September 2006.

[Bau16] Carsten Baum. On garbling schemes with and without privacy. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 468–485. Springer, Cham, August / September 2016.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[BB16] Marina Blanton and Fattaneh Bayatbabolghani. Efficient server-aided secure two-party function evaluation with applications to genomic computation. *PoPETs*, 2016(4):144–164, October 2016.

[BBC+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Cham, August 2019.

[BBC+23] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Arithmetic sketching. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 171–202. Springer, Cham, August 2023.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Berlin, Heidelberg, August 2004.

[BCPS25] Richard Barnes, David Cook, Christopher Patton, and Phillipp Schoppmann. Verifiable Distributed Aggregation Functions. Internet-Draft draft-irtf-cfrg-vdaf-15, Internet Engineering Task Force, June 2025. Work in Progress.

[BF20] Balthazar Bauer and Georg Fuchsbauer. Efficient signatures on randomizable ciphertexts. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 359–381. Springer, Cham, September 2020.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Cham, August 2020.

[BFR24] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On security proofs of existing equivalence class signature schemes. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 3–37. Springer, Singapore, December 2024.

[BJ18] Marina Blanton and Myoungin Jeong. Improved signature schemes for secure multi-party computation with certified inputs. In Javier López, Jianying Zhou, and Miguel Soriano, editors, *ESORICS 2018, Part II*, volume 11099 of *LNCS*, pages 438–460. Springer, Cham, September 2018.

[BMW+18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In *USENIX Security Symposium*, pages 991–1008. USENIX Association, 2018.

[BS20] Dan Boneh and Victor Shoup. A graduate course in applied cryptography (2020). *A book in preparation, v0*, 5:80, 2020.

[CB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *NSDI*, pages 259–282. USENIX Association, 2017.

[CDL16] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *TRUST*, volume 9824 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2016.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 201–210. ACM Press, October / November 2006.

[CHLT25] Rutchathon Chairattana-Apirom, Franklin Harding, Anna Lysyanskaya, and Stefano Tessaro. Server-aided anonymous credentials. *IACR Cryptol. ePrint Arch.*, page 513, 2025.

[CJK+23] Benjamin M. Case, Richa Jain, Alex Koshelev, Andy Leiserson, Daniel Masny, Ben Savage, Erik Taubeneck, Martin Thomson, and Taiki Yamaguchi. Interoperable private attribution: A distributed attribution and aggregation protocol. *IACR Cryptol. ePrint Arch.*, page 437, 2023.

[CKK+25] Elizabeth Crites, Jonathan Katz, Chelsea Komlo, Stefano Tessaro, and Chenzhi Zhu. On the adaptive security of FROST. Cryptology ePrint Archive, Paper 2025/1061, 2025.

[CKL+16] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 3–24. Springer, Cham, August 2016.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Berlin, Heidelberg, September 2003.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Berlin, Heidelberg, August 2004.

[CL19] Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 535–555. Springer, Cham, March 2019.

[CLPP25] Carlos Cela, Ruchi Lohani, Martin Pál, and Chanda Patel. Aggregation Service for the Attribution Reporting API. Technical report, Google, January 2025.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Berlin, Heidelberg, April 1991.

[dCP22] Leo de Castro and Antigoni Polychroniadou. Lightweight, maliciously secure verifiable function secret sharing. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 150–179. Springer, Cham, May / June 2022.

[DDKT25] Nicolas Desmoulins, Antoine Dumanois, Seyni Kane, and Jacques Traoré. Making BBS anonymous credentials eidas 2.0 compliant. *IACR Cryptol. ePrint Arch.*, page 619, 2025.

[DGPS24] Moumita Dutta, Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Compute, but verify: Efficient multiparty computation over authenticated inputs. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VI*, volume 15489 of *LNCS*, pages 133–166. Springer, Singapore, December 2024.

[DJW23] Frank Denis, Frederic Jacobs, and Christopher A. Wood. RSA Blind Signatures. RFC 9474, October 2023.

[DPRS23] Hannah Davis, Christopher Patton, Mike Rosulek, and Phillipp Schoppmann. Verifiable distributed aggregation functions. *Proc. Priv. Enhancing Technol.*, 2023(4):578–592, 2023.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.

[GESM17] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. Cache attacks on intel SGX. In *EUROSEC*, pages 2:1–2:6. ACM, 2017.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 640–658. Springer, Berlin, Heidelberg, May 2014.

[GPP+25] Tim Geoghegan, Christopher Patton, Brandon Pitman, Eric Rescorla, and Christopher A. Wood. Distributed Aggregation Protocol for Privacy Preserving Measurement. Internet-Draft draft-ietf-ppm-dap-14, Internet Engineering Task Force, February 2025. Work in Progress.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Berlin, Heidelberg, December 2007.

[HDRL25] Charlie Harrison, John Delaney, Mariana Raykova, and Nan Lin. Attribution Reporting API with Aggregatable Reports. Technical report, Google, January 2025.

[HIJ+21] Sharon Huang, Subodh Iyengar, Sundar Jeyaraman, Shiv Kushwah, Chen-Kuei Lee, Zutian Luo, Payman Mohassel, Ananth Raghunathan, Shaahid Shaikh, Yen-Chieh Sung, and Albert Zhang. DIT: De-Identified Authenticated Telemetry at Scale. Technical report, April 2021.

[HW25] Scott Hendrickson and Christopher A. Wood. Privacy Pass Token Expiration Extension. Internet-Draft draft-hendrickson-privacypass-expiration-extension-03, Internet Engineering Task Force, January 2025. Work in Progress.

[IT19] Subodh Iyengar and Erik Taubeneck. Fraud Resistant, Privacy Preserving Reporting Using Blind Signatures. Technical report, August 2019.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In Burton S. Kaliski, Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164. Springer, Berlin, Heidelberg, August 1997.

[JT20] Joseph Jaeger and Stefano Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 414–443. Springer, Cham, November 2020.

[KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020.

[KMW16] Jonathan Katz, Alex J. Malozemoff, and Xiao Wang. Efficiently enforcing input validity in secure two-party computation. Cryptology ePrint Archive, Report 2016/184, 2016.

[lib] Divviup `libprio-rs`. https://github.com/divviup/libprio-rs.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Berlin, Heidelberg, March 2012.

[LKWL24] Tobias Looker, Vasilis Kalos, Andrew Whitehead, and Mike Lodder. The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-07, Internet Engineering Task Force, September 2024. Work in Progress.

[LZW+21] Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. CIPHERLEAKS: breaking constant-time cryptography on AMD SEV via the ciphertext side channel. In *USENIX Security Symposium*, pages 717–732. USENIX Association, 2021.

[Mas25] Daniel Masny. Privacy preserving aggregation of ad conversions using mpc. Invited talk, Simons Institute Secure Computation Workshop, August 2025. https://simons.berkeley.edu/talks/daniel-masny-meta-2025-08-08.

[MST24] Dimitris Mouris, Pratik Sarkar, and Nektarios Georgios Tsoutsos. PLASMA: private, lightweight aggregated statistics against malicious adversaries. *Proc. Priv. Enhancing Technol.*, 2024(3):4–24, 2024.

[Orr24] Michele Orrù. Revisiting keyed-verification anonymous credentials. *IACR Cryptol. ePrint Arch.*, page 1552, 2024.

[PLS+25] Andrew Paseltiner, Andy Leiserson, Benjamin Savage, Charlie Harrison, and Martin Thomson. Privacy-Preserving Attribution: Level 1. First-public-working-draft, World Wide Web Consortium, April 2025. Work in Progress.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Cham, February / March 2016.

[ROCT24] Guy N. Rothblum, Eran Omri, Junye Chen, and Kunal Talwar. PINE: efficient verification of a euclidean norm bound of a secret-shared vector. In *USENIX Security Symposium*. USENIX Association, 2024.

[RZCP24] Mayank Rathee, Yuwen Zhang, Henry Corrigan-Gibbs, and Raluca Ada Popa. Private analytics via streaming, sketching, and silently verifiable proofs. In *SP*, pages 3072–3090. IEEE, 2024.

[TC25] Martin Thomson and David Cook. A Prio Instantiation for Vector Sums with an L1 Norm Bound on Contributions. Internet-Draft draft-thomson-ppm-l1-bound-sum-01, Internet Engineering Task Force, June 2025. Work in Progress.

[TCR+22] Nirvan Tyagi, Sofía Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A. Wood. A fast and simple partially oblivious PRF, with applications. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 674–705. Springer, Cham, May / June 2022.

[Tho22] Martin Thomson. An Analysis of Apple's Private Click Measurement. Technical report, June 2022.

[TKM+24] Pierre Tholoniat, Kelly Kostopoulou, Peter McNeely, Prabhpreet Singh Sodhi, Anirudh Varanasi, Benjamin Case, Asaf Cidon, Roxana Geambasu, and Mathias Lécuyer. Cookie monster: Efficient on-device budgeting for differentially-private ad-measurement systems. In *SOSP*, pages 693–708. ACM, 2024.

[TLMR22] Nirvan Tyagi, Julia Len, Ian Miers, and Thomas Ristenpart. Orca: Blocklisting in sender-anonymous messaging. In *USENIX Security Symposium*, pages 2299–2316. USENIX Association, 2022.

[TTS+23] Erik Taubeneck, Martin Thomson, Benjamin Savage, Benjamin Case, Daniel Masny, and Richa Jain. Interoperable Private Attribution End-to-End Protocol. Technical report, Meta and Mozilla, May 2023.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 691–721. Springer, Cham, April 2023.

[Wil21a] John Wilander. Consider using blinded signatures for fraud prevention, April 2021.

[Wil21b] John Wilander. Private Click Measurement. Technical report, Apple, April 2021.

[Win23] Lucas Winstrom. Private Ad Measurement. Technical report, Apple, August 2023.

[YW22] Kang Yang and Xiao Wang. Non-interactive zero-knowledge proofs to multiple verifiers. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 517–546. Springer, Cham, December 2022.

[YW25] Cathie Yun and Christopher A. Wood. Anonymous Rate-Limited Credentials. Internet-Draft draft-yun-cfrg-arc-00, Internet Engineering Task Force, February 2025. Work in Progress.

[ZBB17] Yihua Zhang, Marina Blanton, and Fattaneh Bayatbabolghani. Enforcing input correctness via certification in garbled circuit evaluation. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017, Part II*, volume 10493 of *LNCS*, pages 552–569. Springer, Cham, September 2017.

[ZSS+16] Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y. Thomas Hou. Truspy: Cache side-channel information leakage from the secure world on ARM devices. *IACR Cryptol. ePrint Arch.*, page 980, 2016.

Game $\text{SUF-BBS}^{\mathcal{A}}_{\mathsf{GGen},\ell}(\lambda)$:	Oracle $\mathrm{S}(\vec{m})$:
$\mathsf{Sigs} \leftarrow \emptyset; \mathsf{pp} \leftarrow\!\!\$\ \mathsf{BBS.Setup}(1^\lambda)$	$(A, e) \leftarrow\!\!\$\ \mathsf{BBS.Sign}(\mathsf{pp}, \mathsf{sk}, \vec{m})$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{BBS.KG}()$	$\mathsf{Sigs} \leftarrow \mathsf{Sigs} \cup \{(\vec{m}, A, e)\}$
$(\vec{m}^*, A^*, e^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{S}}(\mathsf{pp}, \mathsf{pk})$	**return** (A, e)
return $((\vec{m}^*, A^*, e^*) \notin \mathsf{Sigs} \wedge$	
$\quad \mathsf{BBS.Ver}(\mathsf{pp}, \mathsf{pk}, \vec{m}^*, (A^*, e^*)) = 1) \vee$	
$\quad A^* = (x+e)^{-1}(\sum_{i=1}^n m_i H_i)$	

Figure 14: Augmented unforgability game for BBS signatures $\mathsf{BBS} = \mathsf{BBS}[\mathsf{GGen}, \ell]$

A Technical Details for BBS-Based SSA

In this section, we give omitted technical details regarding the construction $\mathsf{SSA}_{\mathsf{BBS}}$ based on BBS signatures. In particular, we give description of the proof systems used and the security proofs.

A.1 Additional Security Notion for BBS

We consider an additional security notion for BBS which augments the natural SUF security. In particular, we consider an adversary $\mathcal{A}$ with access to BBS signing oracle but can either produce a well-formed forgery (A, e) or a pair $(\widetilde{A}, e)$ and a message $\vec{m}$ where $\widetilde{A} = (x+e)^{-1}(\sum_{i=1}^n m_i H_i)$. We formalize the game SUF-BBS in Figure 14. The corresponding advantage is defined as

$$\mathsf{Adv}^{\mathsf{suf\text{-}bbs}}_{\mathsf{GGen},\ell}(\mathcal{A}, \lambda) := \Pr[\text{SUF-BBS}^{\mathcal{A}}_{\mathsf{GGen},\ell}(\lambda) = 1] \,.$$

Lemma 5 (Augmented security of BBS)**.** Let GGen be a bilinear group parameter generator outputting groups of prime-order $p = p(\lambda)$ and $\ell = \ell(\lambda) \geqslant 1$ be an integer. For any adversary $\mathcal{A}$ playing the SUF-BBS game and making q signing queries, there exist adversaries $\mathcal{B}_1, \mathcal{B}_2$, and $\mathcal{B}_3$ such that

$$\mathsf{Adv}^{\mathsf{suf\text{-}bbs}}_{\mathsf{GGen},\ell}(\mathcal{A}, \lambda) \leqslant q \cdot \mathsf{Adv}^{q\text{-SDH}}_{\mathsf{GGen}}(\mathcal{B}_1, \lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GGen}}(\mathcal{B}_2, \lambda) + \mathsf{Adv}^{q\text{-SDH}}_{\mathsf{GGen}}(\mathcal{B}_3, \lambda) + \frac{q^2}{2p} + \frac{q+2}{p} \,.$$

The adversaries $\mathcal{B}_1, \mathcal{B}_2$, and $\mathcal{B}_3$ run in time roughly that of $\mathcal{A}$.

Sketch. The proof follows similar argument from [TZ23, Theorem 1] with small changes made to accommodate the additional winning condition. Note that their proof considers three cases: (1) Forgery contains $e^* = e_i$ for some $i \in [q]$ and $A^* \neq A_i$, (2) Forgery contains the same signature as one that was issued but with different message $\vec{m}^*$, (3) Forgery contains $e^* \neq e_i$ for any $i \in [q]$. Here, the additional type of forgery, $\vec{m}^*$ and (A^*, e^*) such that

$$A^* = (x+e^*)^{-1} \left(\sum_{i=1}^n m_i^* H_i \right) \,,$$

can be considered in two cases: (a) $e^* = e_i$ for some $i \in [q]$, and (b) one where $e^* \neq e_i$ for all $i \in [q]$. A key observation here is that proof strategies for cases (1) and (3) work for (a) and (b), respectively. Thus, we arrive at the same bound as in [TZ23, Theorem 1]. □

A.2 Proof Systems Description

For the proof system Π_{Share} which prove knowledge of the rerandomization factors (Π_{Share}), we will use variants of the proof system Π_{Lin} described in Figure 15. The following theorem then establishes the security of the proof system Π_{Lin} in Figure 15. This follows from Fiat-Shamir transform applying to $\Sigma_{\Pi_{\mathsf{Lin}}}$ (see e.g., Boneh-Shoup [BS20, Chapter 19-20]).

Theorem 6. Π_{Lin} satisfies perfect correctness, zero-knowledge, and soundness in the ROM.

$\Pi_{\mathsf{Lin}}.\mathsf{Prove}^{\mathsf{H}}((M \in \mathbb{G}^{n\times m}, \vec{Y} \in \mathbb{G}^n), \vec{x} \in \mathbb{Z}_p^m)$:	$\Pi_{\mathsf{Lin}}.\mathsf{Ver}^{\mathsf{H}}((M \in \mathbb{G}^{n\times m}, \vec{Y} \in \mathbb{G}^n), \pi)$:
$\vec{r} \leftarrow\!\!\$\ \mathbb{Z}_p^m; \vec{R} \leftarrow M\vec{r}; c \leftarrow \mathsf{H}(M, \vec{Y}, \vec{R})$	$(c, \vec{z}) \leftarrow \pi$
$\vec{z} \leftarrow \vec{r} + c \cdot \vec{x}$	$\vec{R} \leftarrow M\vec{z} - c \cdot \vec{Y}$
return $\pi := (c, \vec{z})$	**return** $\mathsf{H}(M, \vec{Y}, \vec{R}) = c$

Figure 15: NIZK proof system $\Pi_{\mathsf{Lin}} = \Pi_{\mathsf{Lin}}[\mathsf{H}, \mathbb{G}]$ for $\mathsf{R}_{\mathbb{G}} := \{((M \in \mathbb{G}^{n\times m}, \vec{Y} \in \mathbb{G}^n), \vec{x} \in \mathbb{Z}_p^m) : \vec{Y} = M\vec{x}\}$.

We will additionally assume that the proof system is straightline-extractable knowledge sound. This property is satisfied in the AGM [FKL18] (see e.g., [Orr24]). Note however that there is a distinction in proving the whole system of $\mathsf{SSA}_{\mathsf{BBS}}$ in the AGM and proving it by (heuristically) assuming that the proof system is straightline-extractable. This is because the group elements that the adversary sees in the whole system and the group elements it sees in the straightline-extractable game may be different.

PROOF SYSTEM Π_{Share}. Recall the corresponding relation

$$\mathsf{R}_{\mathsf{Share}} := \left\{ \begin{array}{c} ((G, \vec{H}, \widetilde{A}, \widetilde{B}, (C_i)_{i\in[n]}, \mathsf{info}); \\ (\alpha, e, \vec{v} \in \mathbb{Z}_p^m, \vec{s} \in (\mathbb{Z}_p^m)^n, \vec{r} \in \mathbb{Z}_p^n)) \end{array} : \begin{array}{l} \widetilde{B} = \alpha(G + \sum_{j=1}^m v_j H_j + \mathsf{info} H_{m+1}) - e\widetilde{A} \wedge \\ \vec{v} = \sum_{i=1}^n \vec{s}_i \wedge \\ \forall i \in [n] : \alpha C_i = \alpha r_i G + \alpha \sum_{j=1}^m s_{i,j} H_j \end{array} \right\}.$$

Note that the first two constraints in the relation can be combined to the following:

$$\widetilde{B} = \alpha \left(G + \sum_{j=1}^m (\sum_{i=1}^n s_{i,j}) H_j + \mathsf{info} H_{m+1} \right) - e\widetilde{A}$$

We define Π_{Share} for a statement $(G, \vec{H}, \widetilde{A}, \widetilde{B}, (C_i)_{i\in[n]}, \mathsf{info})$ and witness $(\alpha, e, \vec{s}, \vec{r})$ as running Π_{Lin} for the following matrix M_{Share} (omitted cells are $0_{\mathbb{G}}$), group elements $\vec{Y}_{\mathsf{Share}}$ and witness $\vec{x}_{\mathsf{Share}} \in \mathbb{Z}_p^{n(m+1)+2}$:

$$M_{\mathsf{Share}} := \begin{pmatrix} G + \mathsf{info} H_{m+1} & -\widetilde{A} & H_1 & \cdots & H_m & \ldots & H_1 & \cdots & H_m & & \\ -C_1 & & H_1 & \ldots & H_m & & & & & G & \\ \vdots & & & & & \ddots & & & & & \ddots \\ -C_n & & & & & & H_1 & \ldots & H_m & & G \end{pmatrix},$$

$$\vec{Y}_{\mathsf{Share}} := \begin{pmatrix} \widetilde{B} \\ \vec{0}_{\mathbb{G}} \end{pmatrix}, \vec{x}_{\mathsf{Share}} := \begin{pmatrix} \alpha \\ e \\ \alpha\vec{s}_1 \in \mathbb{Z}_p^m \\ \vdots \\ \alpha\vec{s}_n \in \mathbb{Z}_p^m \\ \alpha\vec{r} \in \mathbb{Z}_p^n \end{pmatrix}.$$

The extracted witness is in the following relaxed relation:

$$\widetilde{\mathsf{R}}_{\mathsf{Share}} := \left\{ (G, \vec{H}, \widetilde{A}, \widetilde{B}, (C_i)_{i\in[n]}, \mathsf{info}); (\boxed{\alpha}, e, \vec{s}', \vec{r}') \ : \begin{array}{l} \widetilde{B} = \boxed{\alpha}(G + \mathsf{info} H_{m+1}) + \sum_{i=1}^n \sum_{j=1}^m s'_{i,j} H_j - e\widetilde{A} \wedge \\ \forall i \in [n] : \boxed{\alpha} C_i = r'_i G + \sum_{j=1}^m s'_{i,j} H_j \end{array} \right\}.$$

In particular, there is no guarantee that $\alpha \neq 0$. If it is, we can easily extract a BBS signature. When it is not, however, we have that $\widetilde{B} = \sum_{j=1}^m \sum_{i=1}^n s'_{i,j} H_j - e\widetilde{A}$. Since the verification VerPub in SSA $\mathsf{e}(\widetilde{B}, G) = \mathsf{e}(\widetilde{A}, \widehat{X})$ guarantees that $\widetilde{B} = x\widetilde{A}$, we can conclude that $\widetilde{A} = \frac{1}{x+e} (\sum_{i=1}^n s'_i H_i)$. This is not exactly a BBS signature but it can be proved that such group element and scalar pair $(\widetilde{A}, e)$ cannot be produced by an adversary with access to BBS signing oracle as defined in Appendix A.1.

A.3 Security Proofs of BBS-Based SSA

In this section, we give the proof of Theorem 2.

CORRECTNESS. Correctness follows from correctness of BBS signatures and the proof systems.

UNFORGEABILITY. The following lemma establishes the security of $\mathsf{SSA}_{\mathsf{BBS}}$ from unforgeability of BBS signatures and straightline knowledge-soundness of the proof systems $\Pi_{\mathsf{com}}, \Pi_{\mathsf{Share}}$.

Lemma 7. Let GGen be a bilinear group parameter generator outputting groups of prime-order $p = p(\lambda)$ and $\mathsf{SSA}_{\mathsf{BBS}} = \mathsf{SSA}_{\mathsf{BBS}}[\mathsf{GGen}, n, m]$. Then, for any adversary $\mathcal{A}$ (making at most Q_{Iss} issuance query, Q_{H} queries to the random oracles H, and running in time $t_{\mathcal{A}}$), there exist adversaries $\mathcal{B}_{\mathsf{Share}}, \mathcal{B}_{\mathsf{BBS}}, \mathcal{B}_{\mathrm{dlog}}$ running in time roughly that of $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SSA}_{\mathsf{BBS}},n}(\mathcal{A},\lambda) \leqslant \mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{Share}}}(\mathcal{B}_{\mathsf{Share}},\lambda) + \mathsf{Adv}^{\mathsf{suf\text{-}bbs}}_{\mathsf{GGen}}(\mathcal{B}_{\mathsf{BBS}},\lambda) + 2\cdot\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}},\lambda) + \frac{2}{p}\,.$$

Moreover, $\mathcal{B}_{\mathsf{Share}}$ makes at most Q_{H} queries to H and 1 query to the extraction check oracle, and $\mathcal{B}_{\mathsf{BBS}}$ makes at most Q_{Iss} queries to its signing oracle.

Proof. Consider an adversary $\mathcal{A}$ making Q_{Iss} issuance query, Q_{H} queries to the random oracles H, respectively. We assume without loss of generality that the queries to made by the game for verifying the proofs at issuance or at the end of the game are already made by $\mathcal{A}$ in advance. This increase the query count by 1 for H. For each issuance oracle query, we denote the input parsed from the predicate as $(\vec{v}_i \in \mathbb{Z}_p^m, \mathsf{info}_i)$ for $i \in [Q_{\mathrm{Iss}}]$.

Game $\mathbf{G}_0$: This is exactly the unforgeability game.

Game $\mathbf{G}_1$: In this game, we apply the straight-line extractor $\mathsf{Ext}_{\mathsf{Share}}$ for Π_{Share} to the forgery (if it is valid) which is of the following form:

$$(\phi^* = \mathsf{info}^*, \tau^* = (\widetilde{A}^*, \widetilde{B}^*, (C_i^*)_{i\in[n]}, \pi^*_{\mathsf{Share}}), (\mathsf{ss}_i = (\mathsf{info}^*, \vec{s}_i^*), \tau_i = r_i^*))$$

This allows the game to extract the witness $(\alpha, e, \vec{s}' \in (\mathbb{Z}_p^m)^n, \vec{r}' \in \mathbb{Z}_p^n)$ for the relaxed relation $\widetilde{\mathsf{R}}_{\mathsf{Share}}$. Then, by the knowledge-soundness of Π_{Share}, we can see that there exists an adversary $\mathcal{B}_{\mathsf{Share}}$ running in time roughly that of $\mathcal{A}$ making at most Q_{H} queries to H, such that

$$\mathsf{Pr}[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1] \geqslant \mathsf{Pr}[\mathbf{G}_0^{\mathcal{A}}(\lambda) = 1] - \mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{Share}},\mathsf{Ext}_{\mathsf{Share}}}(\mathcal{B}_{\mathsf{Share}},\lambda)\,.$$

Game $\mathbf{G}_2$: In this game, the game aborts if $\alpha \neq 0$ but $\alpha^{-1}\vec{s}' \neq \vec{s}^*$. In this case,, we have that $C_i = r_i^* G + \sum_{j=1}^m s_{i,j}^* H_j = \alpha^{-1} r_i' G + \alpha^{-1}\sum_{j=1}^m s_{i,j}' H_j$ for some $i \in [n]$, breaking discrete logarithm. Hence, following the result in [JT20, Lemma 3] we can construct a reduction $\mathcal{B}_{\mathrm{dlog}}$ running in time roughly that of $\mathcal{A}$ such that

$$\mathsf{Pr}[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] \geqslant \mathsf{Pr}[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1] - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}},\lambda) - \frac{1}{p}\,.$$

Finally, we bound the probability of the adversary winning in game $\mathbf{G}_2$ which consists of the case where the forgery is valid and (1) $\alpha = 0$ or (2) $\alpha \neq 0$ but the commitments and openings correspond to the extracted values, i.e., $\vec{s}^* = \alpha^{-1}\vec{s}'$ and $\vec{r}^* = \alpha^{-1}\vec{r}'$. For case (1), one can derive that $\widetilde{A}^* = (x+e)^{-1}\left(\sum_{i=1}^n s_i' H_i\right)$, which is a forgery in the augmented security game of BBS. For (2), since the forgery is valid and fresh, a valid signature for $(\vec{v}^* = \sum_{i=1}^n \vec{s}_i^*, \mathsf{info}^*)$ has not been queried before. We can derive the group element

$$A^* = \alpha'^{-1}\widetilde{A}^* = \frac{\alpha'^{-1}}{x+e}\left(\alpha(G + \mathsf{info}H_{n+1}) + \sum_{j=1}^m\sum_{i=1}^n s_{i,j}' H_j\right) = \frac{G + \sum_{j=1}^m v_j^* H_j + \mathsf{info}^* H_{n+1}}{x+e}\,.$$

Therefore, this is a fresh BBS signature on $(\vec{v}^*, \mathsf{info}^*)$. Accordingly, we can construct an adversary $\mathcal{B}_{\mathsf{BBS}}$ for the augmented unforgeability game of BBS signatures SUF-BBS. The adversary $\mathcal{B}_{\mathsf{BBS}}$ will (a) simulate the issuance using the signing oracle on the input $(\vec{v}_i, \mathsf{info}_i)$ (hence, making at most Q_{Iss} signing queries) and (b) depending on whether (1) or (2) occur, it will compute a forgery accordingly. The correctness of such reduction follows from the discussion above. Hence,

$$\mathsf{Pr}[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] \leqslant \mathsf{Adv}^{\mathsf{suf\text{-}bbs}}_{\mathsf{GGen}}(\mathcal{B}_{\mathsf{BBS}},\lambda)\,.$$

□

UNLINKABILITY. Unlinkability of the scheme follows from

Lemma 8. Let GGen be a bilinear group parameter generator outputting groups of prime-order $p = p(\lambda)$ and $\mathsf{SSA}_{\mathsf{BBS}} = \mathsf{SSA}_{\mathsf{BBS}}[\mathsf{GGen}, n, m]$. Then, for any adversary $\mathcal{A}$ (making at most Q_{Share} queries to Share and Q_{H} queries to the random oracle H), there exist an adversary $\mathcal{B}_{\mathsf{Share}}$ such that

$$\mathsf{Adv}^{\mathsf{unlk}}_{\mathsf{SSA}_{\mathsf{BBS}},n}(\mathcal{A},\lambda) \leqslant 2 \cdot \mathsf{Adv}^{\mathsf{zk}}_{\Pi_{\mathsf{Share}}}(\mathcal{B}_{\mathsf{Share}},\lambda) \; .$$

Moreover, $\mathcal{B}_{\mathsf{Share}}$ makes at most Q_{H} queries to H and Q_{Share} queries to its prover oracle.

Proof. Consider an adversary $\mathcal{A}$ playing the unlinkability game, picking a possibly malicious public key $\mathsf{pk} = \widehat{X}$, invoking user-side oracles $\mathrm{U}_1, \mathrm{U}_2$ for $\beta = 0, 1$ (possibly issuing maliciously formed credentials), and querying the share oracle Share at most Q_{Share} times. Note that in U_2, the game checks whether the issued credential is valid or not, and in Share, the adversary only learns $n-1$ out of n shares. Also, by the condition in Share that the specified predicate ϕ should be satisfied by both inputs v_0, v_1, so we assume w.l.o.g. that v_0, v_1 contains the same public tag $\mathsf{info}_0 = \mathsf{info}_1$ (otherwise, $\mathcal{A}$ learns nothing). Now, we consider the following sequence of games parameterized by an index i and a bit b–the bit denotes which bit b is used in the unlinkability game.

Game $\mathbf{G}_{0,b}$: This is exactly the game $\mathrm{UNLK}_{\mathsf{SSA}_{\mathsf{BBS}},n,b}$.

Game $\mathbf{G}_{1,b}$: In this game, the game simulate the proofs π_{Share} in the Share using the simulator $\mathsf{Sim}_{\mathsf{Share}}$ implied by the zero-knowledge property of Π_{Share}. Here, we can construct an adversary $\mathcal{B}_{\mathsf{Share}}$ making Q_{H} to H and Q_{Share} queries to the prover oracle, such that

$$|\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{0,b}(\lambda) = 1] - \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{1,b}(\lambda) = 1]| \leqslant \mathsf{Adv}^{\mathsf{zk}}_{\Pi_{\mathsf{Share}}}(\mathcal{B}_{\mathsf{Share}},\lambda) \; .$$

Game $\mathbf{G}_{2,b}$: On each query to Share, the group elements $\widetilde{A}, \widetilde{B}$ are computed as $(\widetilde{A}, \widetilde{B}) \leftarrow \alpha(G, X)$ where $\alpha \leftarrow\!\!\$\, \mathbb{Z}_p^*$. Note that this does not need to be efficient as we are targetting unbounded adversaries here, meaning the game can simply brute-force the secret key x of $\mathsf{pk} = \widehat{X}$. This is indistinguishable to $\mathbf{G}_{1,b}$ since the rerandomization factors are not used (because the proofs are simulated) and $(\widetilde{A}, \widetilde{B})$ are always a Diffie-Hellman tuple with respect to the public key (G, X) as the credentials used to generated them in $\mathbf{G}_2$ are guaranteed to be valid (otherwise the oracle abort). Hence,

$$\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{2,b}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{1,b}(\lambda) = 1] \; .$$

Game $\mathbf{G}_{3,b}$: On each query to Share with the subset $S \subsetneq [n]$, for each $i \in [n] \backslash S$ be the non-opened index and compute $C_{i*} \leftarrow\!\!\$\, r'G$ for $r' \leftarrow\!\!\$\, \mathbb{Z}_p$ instead. Since the oracle does not open C_{i*}, this is indistinguishable by perfectly hiding property of Pedersen commitment (note that the non-opened openings are also not needed anymore as the proofs are simulated). Hence,

$$\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{3,b}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{2,b}(\lambda) = 1] \; .$$

Game $\mathbf{G}_{4,b}$: Now, in each Share oracle, instead of computing s_i for $i \neq i^*$ as secret shares of v_0 or v_1, we compute them as secret shares of 0. Since only $n-1$ of these shares are revealed to the adversary, no information about the secret is revealed to the adversary (i.e., perfect privacy of the secret sharing scheme), this game is identical to $\mathbf{G}_{3,b}$. Hence,

$$\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{3,b}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{3,b}(\lambda) = 1] \; .$$

Now, $\mathbf{G}_{4,0} = \mathbf{G}_{4,1}$ as nothing depends on the attributes and everything is simulated, meaning we have proved the lemma statement. □

B Security Proof of the SEQ Scheme

In this section, we give a proof of Theorem 3 for general $m \geqslant 1$. We will show that adaption (Appendix B.1) and unforgeability (Appendix B.2) are satisfied. Note that correctness and perfect hiding are trivially satisfied–the latter follows from perfect hiding of Pedersen commitments.

B.1 Adaption

We state and prove the following lemma

Lemma 9. Let GGen be a bilinear group parameter generator outputting groups of prime-order $p = p(\lambda), n = n(\lambda), m = m(\lambda)$ be integers, $A \in \mathbb{Z}_p^{\ell \times n}$ be a full-row-rank matrix, and $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, n, m, A]$. Then, for any adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{adapt}}_{\mathsf{SEQ}}(\mathcal{A}, \lambda) := 0$$

Fix an arbitrary public key $\mathsf{pk} = (X_1, \ldots, X_n) \in \widehat{\mathbb{G}}^n$, messages $\vec{\mu} \in (\mathbb{Z}_p^m)^n$, randomness $\vec{r} \in \mathbb{Z}_p^n$, commitments $\vec{C} = \mathsf{SEQ.Com}(\mathsf{pp}, \vec{\mu}; \vec{r})$, and signature $\sigma = (Z, (T_{j',j})_{j' \in [m], j \in [\ell]}, (\overline{T}_k)_{k \in [n]}, S, \widehat{S})$ such that $\mathsf{SEQ.Ver}(\mathsf{pp}, \mathsf{pk}, \vec{C}, \sigma) = 1$. This means that:

$$\begin{aligned}
S &\neq 0_{\mathbb{G}} \\
\mathsf{e}(S, \widehat{G}) &= \mathsf{e}(G, \widehat{S}) \\
\mathsf{e}(Z, \widehat{S}) &= \mathsf{e}(G, \widehat{G}) + \sum_{i=1}^{n} \mathsf{e}(C_i, X_i) \\
\forall j' \in [m], j \in [\ell] : \mathsf{e}(T_j, \widehat{S}) &= \sum_{i=1}^{n} A_{j,i} \mathsf{e}(H_{j'}, X_i) \\
\forall k \in [n] : \mathsf{e}(\overline{T}_k, \widehat{S}) &= \mathsf{e}(G, X_i)
\end{aligned}$$

Let $x_i = \mathrm{dlog}_{\widehat{G}} X_i$ for $i \in [n]$. Then, there exists some $s \in \mathbb{Z}_p^*$ such that

$$\begin{aligned}
\mathrm{dlog}_G S = \mathrm{dlog}_{\widehat{G}} \widehat{S} &= s^{-1} \\
Z = s\left(G + \sum_{i=1}^{n} x_i C_i\right) &= s\left(G + \sum_{i=1}^{n} x_i \left(r_i G + \sum_{j'=1}^{m} \mu_{i,j'} H_{j'}\right)\right) \\
\forall j' \in [m], j \in [\ell] : T_{j',j} &= s \sum_{i=1}^{n} A_{j,i} x_i H_{j'} \\
\forall k \in [n] : \overline{T}_k &= s x_i G
\end{aligned}$$

Now, consider any $\vec{\mu}' \in [\![\vec{\mu}]\!]_{\mathsf{pp}}$ and $\vec{r}' \in \mathbb{Z}_p^n$. One can observe that because A is full-row rank, there exists a unique $\vec{\alpha} \in \mathbb{Z}_p^{m\ell}$ such that $\vec{\mu}' = \left(\vec{\mu}_i + \sum_{j=1}^{\ell} A_{j,i} \vec{\alpha}_j\right)_{i \in [n]}$. Moreover, there also exists a fixed $\beta \in \mathbb{Z}_p^n$ such that $\vec{r}' = \vec{r} + \vec{\beta}$. This means that the rerandomized message and randomness from the Adapt algorithm is distributed identically to ones sampled from $[\![\vec{\mu}]\!]_{\mathsf{pp}}$ and $\vec{r}' \leftarrow\!\!\$\, \mathbb{Z}_p^n$. Now, we only have to show that the rerandomized signature σ' is distributed uniformly in the set of valid signatures for $\vec{C}' = \mathsf{SEQ.Com}(\mathsf{pp}, \vec{\mu}'; \vec{r}')$. In particular, let $\mathsf{rnd} = (\vec{\alpha}, \vec{\beta})$ be as described above. Then, consider the output of Adapt, $\sigma' = (Z', S', \widehat{S}')$.

$$\begin{aligned}
S' &= \gamma^{-1} S = \gamma^{-1} s^{-1} G \\
\widehat{S}' &= \gamma^{-1} S' = \gamma^{-1} s^{-1} \widehat{G} \\
Z' &= \gamma \left(Z + \sum_{j'=1}^{m} \sum_{j=1}^{\ell} \alpha_{j,j'} T_{j',j} + \sum_{k=1}^{n} \beta_k \overline{T}_k\right) \\
&= \gamma s \left(G + \sum_{i=1}^{n} x_i \left(r_i G + \sum_{j'=1}^{m} \mu_{i,j'} H_{j'}\right) + \sum_{j'=1}^{m} \sum_{j=1}^{\ell} \alpha_{j,j'} \sum_{i=1}^{n} A_{j,i} x_i H_{j'} + \sum_{k=1}^{n} \beta_k x_k G\right) \\
&= \gamma s \left(G + \sum_{i=1}^{n} x_i \left(\sum_{j'=1}^{m} (m_{i,j'} + \sum_{j=1}^{\ell} \alpha_{j,j'} A_{j,i}\right) H_{j'} + (r_i + \beta_i) G\right)
\end{aligned}$$

Game q-PowDenDL$^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)$:

$(p, \mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, \mathsf{e}) \leftarrow\!\$\ \mathsf{GGen}(1^\lambda); G \leftarrow \mathbb{G}^*; \widehat{G} \leftarrow\!\$\ \widehat{\mathbb{G}}^*$

$\mathsf{pp} = ((p, \mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, \mathsf{e}), G, \widehat{G})$

$y, c_1, \ldots, c_q \leftarrow\!\$\ \mathbb{Z}_p$

if $y \in \{-c_1, \ldots, -c_q\}$ **then return** 1

$y' \leftarrow\!\$\ \mathcal{A}(\mathsf{pp}, (y^i G)_{i \in [2q]}, y\widehat{G}, (\frac{1}{y+c_i} G, \frac{1}{y+c_i} \widehat{G}, c_i)_{i \in [q]})$

return $y = y'$

Figure 16: Game q-PowDenDL.

$$= \gamma s \left(G + \sum_{i=1}^{n} x_i C'_i \right)$$

Here, $\gamma \in \mathbb{Z}_p^*$ is the sampled randomness of Adapt. Note that this adapted signature verifies (via $\mathsf{SEQ.VerAdapt}$) for $\vec{C}'$, and the distribution is identical, since if S' is fixed, then $Z', \widehat{S}'$ are also fixed. Moreover, S' is distributed uniformly in $\mathbb{G}^*$. Hence, we proved adaption.

B.2 Unforgeability

We state and prove the following lemma.

Lemma 10. Let GGen be a bilinear group parameter generator outputting groups of prime-order $p = p(\lambda)$, $n = n(\lambda), m = m(\lambda)$ be integers, $A \in \mathbb{Z}_p^{\ell \times n}$ be a full-row-rank matrix, and $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, n, m, A]$. Then, for any algebraic adversary $\mathcal{A}$ making at most $q = q(\lambda)$ signing queries and running in time $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$, there exists an adversary $\mathcal{B}$ (in the AGM) with running time roughly $t_{\mathcal{A}}$ such that

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SEQ}}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{q\text{-powdendl}}_{\mathsf{GGen}}(\mathcal{B}, \lambda) + \frac{3q+3}{p}\,.$$

Let n be some fixed integer and A be the full-row-rank matrix parameterizing the scheme. In this section, we give a proof for the unforgeability property of the SEQ scheme in the AGM. The proof follows a similar strategy to that of [BFR24] and reduces to the same assumption q-PowDenDL (of which the corresponding game is defined in Figure 16). Note that the assumption is tightly implied by the standard $(3q, q+1) - \mathrm{DL}$ assumption as proved in [BFR24, Lemma 3].

DEFINING VARIABLES IN THE REDUCTION. We consider the adversary $\mathcal{A}$ in the algebraic group model (AGM), i.e., every time that $\mathcal{A}$ outputs group elements, it also outputs their description as linear combinations its group elements inputs. For the sake of analysis, we denote the inputs and outputs of $\mathcal{A}$ using polynomials in variables $\mathsf{X}_1, \ldots, \mathsf{X}_n$ (denoting the secret keys), $\mathsf{H}_1, \ldots, \mathsf{H}_m$ (denoting the discrete logarithm of the public parameters), $\mathsf{S}_1, \ldots, \mathsf{S}_q$ (denoting the randomness during each signing session). Let $\vec{\mu}^{(i)}, \vec{r}^{(i)}$ denote the i-th query to the signing oracle made by $\mathcal{A}$. Now, during the game, we can denote the i-th query outputs $Z^{(i)}, T^{(i)}_{1,1}, \ldots, T^{(i)}_{m,\ell}, \overline{T}^{(i)}_1, \ldots, \overline{T}^{(i)}_n$ with polynomials $\mathsf{Z}^{(i)}, \mathsf{T}^{(i)}_{1,1}, \ldots, \mathsf{T}^{(i)}_{m,\ell}, \overline{\mathsf{T}}^{(i)}_1, \ldots, \overline{\mathsf{T}}^{(i)}_n$ in the variables $(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}})$. In particular,

$$\mathsf{Z}^{(i)}(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \mathsf{S}_i \left(1 + \sum_{k=1}^{n} (r^{(i)}_k + \sum_{j=1}^{m} \mu^{(i)}_{k,j} \mathsf{H}_j) \mathsf{X}_k \right),$$

$$\mathsf{T}^{(i)}_{j',j}(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \mathsf{S}_i \left(\sum_{k=1}^{n} A_{j,k} \mathsf{X}_k \right) \mathsf{H}_{j'}$$

$$\overline{\mathsf{T}}^{(i)}_j(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \mathsf{S}_i \mathsf{X}_j$$

Moreover, we can write the outputs $(\vec{\mu}^*, \vec{r}^*, \sigma^* = (Z^*, S^*, \widehat{S}^*))$ of $\mathcal{A}$ at the end of the game as Laurent polynomials

$$\mathsf{Z}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \zeta_g + \sum_{j'=1}^{m} \zeta_{h,j'} \mathsf{H}_{j'} + \sum_{i=1}^{q} \left(\zeta_{s,i} \mathsf{S}_i^{-1} + \zeta_{z,i} \mathsf{Z}^{(i)} + \sum_{j'=1}^{m} \sum_{j=1}^{\ell} \zeta_{t,i,j',j} \mathsf{T}_{j',j}^{(i)} + \sum_{j=1}^{n} \zeta_{\bar{t},i,n} \overline{\mathsf{T}}_j^{(i)} \right)$$

$$\mathsf{S}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \rho_g + \sum_{j'=1}^{m} \rho_{h,j'} \mathsf{H}_{j'} + \sum_{i=1}^{q} \left(\rho_{s,i} \mathsf{S}_i^{-1} + \rho_{z,i} \mathsf{Z}^{(i)} + \sum_{j'=1}^{m} \sum_{j=1}^{\ell} \rho_{t,i,j',j} \mathsf{T}_{j',j}^{(i)} + \sum_{j=1}^{n} \rho_{\bar{t},i,n} \overline{\mathsf{T}}_j^{(i)} \right)$$

$$\widehat{\mathsf{S}}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \widehat{\rho}_g + \sum_{k=0}^{n} \widehat{\rho}_{x,k} \mathsf{X}_k + \sum_{i=1}^{q} \widehat{\rho}_{s,i} \mathsf{S}_i^{-1}$$

REDUCTION STRATEGY. We first describe the reduction $\mathcal{B}$ which plays the q-PowDenDL game and runs an algebraic adversary $\mathcal{A}$ playing the unforgeability game of SEQ. At the high level, the reduction $\mathcal{B}$ on input $(\mathsf{pp}, (Y_i)_{i \in [2q]}, \widehat{Y}_1, (A_i, \widehat{A}_i, c_i)_{i \in [q]})$ does the following:

- **Setting up the keys and parameters:** Sample randomizing factors $r_1, \ldots, r_q, a_1, \ldots, a_n, a'_1, \ldots, a'_m \leftarrow\!\!\$\, \mathbb{Z}_p^*$, and $b_1, \ldots, b_n, b'_1, \ldots, b'_m \leftarrow\!\!\$\, \mathbb{Z}_p$. Set the keys $X_i \leftarrow a_0 \widehat{Y}_1 + b_0 \widehat{G}$ for $i \in [0, n]$ and the public parameters $H_i \leftarrow a'_i Y_1 + b'_i G$. Note that now, we can denote the variables $\mathsf{X}_i, \mathsf{H}_i$ as $\mathsf{X}_i(\mathsf{Y}) = a_i \mathsf{Y} + b_i, \mathsf{H}_i(\mathsf{Y}) = a'_i \mathsf{Y} + b'_i$. Additionally, note that if the input contains c_i such that $-c_i G = Y_1$, then the reduction returns $-c_i$.

- **Answering signing queries:** For each signing query $(\vec{\mu}^{(i)}, \vec{r}^{(i)})$, with how the keys has been setup, the reduction reply to the i-th signing queries using linear combinations of $G, Y_1, \ldots, Y_{2i}, A_i$ and $\widehat{A}_i$. In particular, we set $s_i = r_i(y + c_i)$ where $y = \mathrm{dlog}_G Y_1$. Accordingly, this means denoting the variable S_i as $\mathsf{S}_i(\mathsf{Y}) = r_i(\mathsf{Y} + c_i)$. Then, we have that one can write the polynomials $Z^{(i)}, T^{(i)}_{j',j}, \overline{T}^{(i)}_j$ in the variable Y by substituting $\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}$ with the above, and also the reduction can answer signing queries with the following:

$$S_i = r_i^{-1} A_i \,, \widehat{S}_i = r_i^{-1} \widehat{A}_i \,, Z^{(i)} = \mathsf{Z}^{(i)}(y) G \,, T^{(i)}_{j',j} = \mathsf{T}^{(i)}_{j',j}(y) G \,, \overline{T}^{(i)}_j = \overline{\mathsf{T}}^{(i)}_j(y) G \,.$$

  Note that these group elements can be computed as a linear combination of $\mathcal{B}$'s input, since $\mathsf{Z}^{(i)}(\mathsf{Y}), \mathsf{T}^{(i)}_{j',j}(\mathsf{Y}), \overline{\mathsf{T}}^{(i)}_j(\mathsf{Y})$ are only of degree at most 3 polynomials.

- **Extracting solution:** Finally, $\mathcal{A}$ returns $(\vec{\mu}^*, \vec{r^*}, \sigma^* = (Z^*, S^*, \widehat{S}^*))$ along with the algebraic descriptions $\vec{\zeta}, \vec{\rho}, \vec{\widehat{\rho}}$ of $Z^*, S^*, \widehat{S}^*$, respectively.

  Then, $\mathcal{B}$ interpret Z^*, T^*, S^* as linear combinations of $\mathsf{pk}, (Z^{(i)}, T^{(i)}_{j',j}, \overline{T}^{(i)}_j, S^{(i)}, \widehat{S}^{(i)})$, inducing the rational functions $\mathsf{Z}^*, \mathsf{S}^*, \widehat{\mathsf{S}}^*$ as discussed above. Also, we denote the following polynomials related to the verification equations:

$$\mathsf{V}_1(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \prod_{i=1}^{q} \mathsf{S}_i \left(\mathsf{S}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) - \widehat{\mathsf{S}}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) \right) \,,$$

$$\mathsf{V}_2(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) := \prod_{i=1}^{q} \mathsf{S}_i^2 \left(1 + \sum_{k=1}^{n} \mathsf{X}_k \left(r_k^* + \sum_{j=1}^{m} \vec{\mu}_{k,j}^* \mathsf{H}_j \right) - \mathsf{Z}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) \widehat{S}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) \right) \,.$$

  Then, the reduction check that the adversary $\mathcal{A}$ wins, and that one of the corresponding rational functions for each verification equation are non-zero, i.e., one of the following is true

$$\mathsf{Q}_1(\mathsf{Y}) := \mathsf{V}_1(\vec{a}\mathsf{Y} + \vec{b}, \vec{a}'\mathsf{Y} + \vec{b}', \vec{r}\mathsf{Y} + \vec{r} \circ \vec{c}) \neq 0$$

$$\mathsf{Q}_2(\mathsf{Y}) := \mathsf{V}_1(\vec{a}\mathsf{Y} + \vec{b}, \vec{a}'\mathsf{Y} + \vec{b}', \vec{r}\mathsf{Y} + \vec{r} \circ \vec{c}) \neq 0 \,.$$

  Note that we denote $\vec{a} \circ \vec{b} = (a_1 b_1, \ldots, a_k b_k)$ as element-wise multiplication for vectors $\vec{a}, \vec{b} \in \mathbb{Z}_p^k$. If the check passes, the reduction finds a root y of the non-zero rational function, such that $yG = Y_1$, and returns such y.

We note first that the distribution of the view of $\mathcal{A}$ within the reduction is identical to its view in the UNF game, except for when $y + c_i = 0$ for some $i \in [q]$. This is because X_i and H_i are uniformly random in $\mathbb{G}$ and if $y + c_i \neq 0$, S_i is uniformly random in $\mathbb{G}^*$. The bad event occurs with probability at most q/p.

Now, we define two additional bad events $\mathsf{Bad}_1, \mathsf{Bad}_2$ as follows:

- Bad_1 : The adversary $\mathcal{A}$ wins in the game, but the polynomials V_1 and V_2 are both 0 polynomials.
- Bad_2 : The adversary $\mathcal{A}$ wins in the game, but the event Bad_1 does not occur (i.e., $\mathsf{V}_1 \neq 0$ or $\mathsf{V}_2 \neq 0$) and $\mathsf{Q}_1 = 0$ or $\mathsf{Q}_2 = 0$.

Note that if neither events occur, we have that $\mathsf{Q}_1 \neq 0$ and $\mathsf{Q}_2 \neq 0$, but $\mathsf{Q}_1(y) = 0$ or $\mathsf{Q}_2(y) = 0$. Therefore, $\mathcal{B}$ will find the discrete logarithm y by factoring the polynomials.

ANALYSIS OF EVENT Bad_1. We note again that the event Bad_1 occurring while $\mathcal{A}$ wins the game means that the following equations are satisfied.

$$\widehat{\mathsf{S}}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) = \mathsf{S}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) \neq 0 \tag{5}$$

$$\mathsf{Z}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) \cdot \widehat{\mathsf{S}}^*(\vec{\mathsf{X}}, \vec{\mathsf{H}}, \vec{\mathsf{S}}) = 1 + \sum_{k=1}^{n} \mathsf{X}_k \left(r_k^* + \sum_{j=1}^{m} \vec{\mu}_{k,j}^* \mathsf{H}_j \right) \tag{6}$$

Our goal now is to show that if the above equations are satisfied, then there exists some $i^* \in [q]$ such that $\vec{\mu}^* \in [\vec{\mu}^{(i^*)}]_{\mathsf{pp}}$ by considering the coefficients of these polynomials. Thus, there is a contradiction, since $\mathcal{A}$ wins the game.

Now, we prove the following lemma.

Lemma 11. $\widehat{\mathsf{S}}^*(\vec{\mathsf{X}}, \mathsf{H}, \vec{\mathsf{S}}) = \widehat{\rho}_g + \sum_{i=1}^{q} \widehat{\rho}_{s,i} \mathsf{S}_i^{-1}$

Proof. We will first consider Equation (5) multiplying $\prod_{i=1}^{q} \mathsf{S}_i$ on both sides of the equations. Now, it becomes a polynomial equation (without any rational terms), i.e.,

$$\begin{aligned}
&\prod_{i=1}^{q} \mathsf{S}_i \left(\rho_g + \sum_{j=1}^{m} \rho_{h,j} \mathsf{H}_j + \sum_{i=1}^{q} \left(\rho_{s,i} \mathsf{S}_i^{-1} + \rho_{z,i} \mathsf{Z}^{(i)} + \sum_{j'=1}^{m} \sum_{j=1}^{\ell} \rho_{t,i,j',j} \mathsf{T}_{j',j}^{(i)} + \sum_{j=1}^{n} \rho_{\bar{t},i,n} \overline{\mathsf{T}}_j^{(i)} \right) \right) \\
&\quad = \prod_{i=1}^{q} \mathsf{S}_i \left(\widehat{\rho}_g + \sum_{k=0}^{n} \widehat{\rho}_{x,k} \mathsf{X}_k + \sum_{i=1}^{q} \widehat{\rho}_{s,i} \mathsf{S}_i^{-1} \right)
\end{aligned}$$

We consider the above equation in the quotient ring with respect to the ideal generated by $\{\mathsf{X}_1, \ldots, \mathsf{X}_n\}$, which gives

$$\prod_{i=1}^{q} \mathsf{S}_i (\rho_g + \sum_{j=1}^{m} \rho_{h,j} \mathsf{H}_j + \sum_{i=1}^{q} \rho_{s,i} \mathsf{S}_i^{-1}) = \prod_{i=1}^{q} \mathsf{S}_i (\widehat{\rho}_g + \sum_{i=1}^{q} \widehat{\rho}_{s,i} \mathsf{S}_i^{-1})$$

By equating the coefficients, we have that $\rho_{h,j'} = 0$ for $j \in [m]$, $\rho_g = \widehat{\rho}_g$, and $\rho_{s,i} = \widehat{\rho}_{s,i}$ for all $i \in [q]$. Next, we subtract these equal terms from both sides, which gives

$$\prod_{i=1}^{q} \mathsf{S}_i \sum_{i=1}^{q} \left(\rho_{z,i} \mathsf{Z}^{(i)} + \sum_{j'=1}^{m} \sum_{j=1}^{\ell} \rho_{t,i,j',j} \mathsf{T}_{j',j}^{(i)} + \sum_{j=1}^{n} \rho_{\bar{t},i,n} \overline{\mathsf{T}}_j^{(i)} \right) = \prod_{i=1}^{q} \mathsf{S}_i \sum_{k=0}^{n} \widehat{\rho}_{x,k} \mathsf{X}_k \, .$$

Since there is no $\mathsf{X}_k \prod_{i=1}^{q} \mathsf{S}_i$ monomials in the LHS (any monomial in Z, T and $\overline{\mathsf{T}}$ has $\mathsf{S}_i \mathsf{X}_k$ for some i and k), we have that $\widehat{\rho}_{x,k} = 0$ for all $k \in [0, n]$. This proves the lemma. □

With this new derivation of $\widehat{\mathsf{S}}^*$, we substitute it back into Equation (6) and multiplying by $\prod_{i=1}^{q} \mathsf{S}_i^2$ (make any

monomial not contain S_i^{-1})., giving

$$\prod_{i=1}^{q}\mathsf{S}_i^2\left(\widehat{\rho}_g+\sum_{i=1}^{q}\widehat{\rho}_{s,i}\mathsf{S}_i^{-1}\right)\cdot\left(\zeta_g+\sum_{j'=1}^{m}\zeta_{h,j'}\mathsf{H}_{j'}+\sum_{i=1}^{q}\left(\zeta_{s,i}\mathsf{S}_i^{-1}+\zeta_{z,i}\mathsf{Z}^{(i)}+\sum_{j'=1}^{m}\sum_{j=1}^{\ell}\zeta_{t,i,j',j}\mathsf{T}_{j',j}^{(i)}+\sum_{j=1}^{n}\zeta_{\bar{t},i,n}\overline{\mathsf{T}}_j^{(i)}\right)\right)$$
$$=\prod_{i=1}^{q}\mathsf{S}_i^2\left(1+\sum_{k=1}^{n}\mathsf{X}_k(r_k^*+\sum_{j=1}^{m}\vec{\mu}_{k,j}^*\mathsf{H}_j)\right)\,. \tag{7}$$

We now show the following lemma, which concludes the proof.

Lemma 12. There exists $i^*\in[q]$ such that for all $k\in[n],j'\in[m]$

$$\mu_{k,j'}^*=\mu_{k,j'}^{(i^*)}+\sum_{j=1}^{\ell}\widehat{\rho}_{s,i^*}\zeta_{t,i^*,j',j}A_{j,k}\,,$$
$$r_k^*=r_k^{(i^*)}+\widehat{\rho}_{s,i^*}\zeta_{\bar{t},i^*,k}\,.$$

Proof. We start by inspecting Equation (7). First, consider the equation in the quotient ring with respect to the ideal $\mathcal{J}=\langle\mathsf{X}_1,\ldots,\mathsf{X}_n\rangle$. Since the RHS all have terms divisible by X_k and $\mathsf{Z}^{(i)},\mathsf{T}_j^{(i)}$ are all 0 modulo $\mathcal{J}$, we have

$$\prod_{i=1}^{q}\mathsf{S}_i^2\left(\widehat{\rho}_g+\sum_{i=1}^{q}\widehat{\rho}_{s,i}\mathsf{S}_i^{-1}\right)\left(\zeta_g+\sum_{j'=1}^{m}\zeta_{h,j'}\mathsf{H}_{j'}+\sum_{i=1}^{q}\zeta_{s,i}\mathsf{S}_i^{-1}+\zeta_{z,i}\mathsf{S}_i\right)\equiv\prod_{i=1}^{q}\mathsf{S}_i^2\pmod{\mathcal{J}}\,.$$

Expanding the terms (and dividing $\prod_{i=1}^{q}\mathsf{S}_i^2$ off from both sides), we have

$$1=\widehat{\rho}_g\zeta_g+\sum_{i=1}^{q}\widehat{\rho}_{s,i}\zeta_{z,i}+\sum_{j'=1}^{m}\zeta_{h,j'}\widehat{\rho}_g\mathsf{H}_{j'}+\widehat{\rho}_g\sum_{i=1}^{q}\zeta_{s,i}\mathsf{S}_i+\sum_{i=1}^{q}\left(\zeta_g\widehat{\rho}_{s,i}\mathsf{S}_i^{-1}+\sum_{j'=1}^{m}\zeta_{h,j'}\widehat{\rho}_{s,i}\mathsf{S}_i^{-1}\mathsf{H}_{j'}\right)$$
$$+\sum_{i,i'\in[q]}\widehat{\rho}_{s,i}\zeta_{s,i'}\mathsf{S}_i^{-1}\mathsf{S}_{i'}^{-1}+\sum_{i\neq i'\in[q]}\widehat{\rho}_{s,i}\zeta_{z,i'}\mathsf{S}_i^{-1}\mathsf{S}_{i'}$$

Equating coefficients gives us

$$\widehat{\rho}_g\zeta_g+\sum_{i=1}^{q}\widehat{\rho}_{s,i}\zeta_{z,i}=1\,,$$
$$\zeta_{h,j'}\widehat{\rho}_g=0 \qquad \forall j'\in[m]$$
$$\zeta_g\widehat{\rho}_{s,i}+\widehat{\rho}_g\zeta_{s,i}=0 \qquad \forall i\in[q]$$
$$\zeta_{h,j'}\widehat{\rho}_{s,i}=\zeta_{z,i}\widehat{\rho}_g=0 \qquad \forall j'\in[m],i\in[q]$$
$$\widehat{\rho}_{s,i}\zeta_{s,i}=0\ \wedge\ \widehat{\rho}_{s,i}\zeta_{s,i'}+\widehat{\rho}_{s,i'}\zeta_{s,i}=0\ \wedge\ \widehat{\rho}_{s,i}\zeta_{z,i'}=0 \qquad \forall i\neq i'\in[q]$$

Note that $\zeta_{h,j'}=0$ for all $j'\in[m]$. Otherwise, $\widehat{\rho}_g=\widehat{\rho}_{s,i}=0$ for all $i\in[q]$, which is a contradiction as $\widehat{\mathsf{S}}^*\neq 0$.

Next, assume that there exists i' such that $\zeta_{s,i'}\neq 0$. Then, $\widehat{\rho}_{s,i'}=0$, and consequently, $\widehat{\rho}_g=\widehat{\rho}_{s,i}=0$ for all $i\in[q]\backslash\{i'\}$, which is a contradiction to $\widehat{\mathsf{S}}^*\neq 0$. Hence $\zeta_{s,i}=0$ for all $i\in[q]$. Similarly, $\zeta_g=0$.

Moreover, assume there exists $i'\neq i''$ such that $\zeta_{z,i'},\zeta_{z,i''}\neq 0$, then $\widehat{\rho}_g=\widehat{\rho}_{s,i}=0$ for all $i\in[q]$, a contradiction, meaning there exists a unique $i^*\in[q]$ such that $\zeta_{z,i^*}\neq 0$.

With this, we also have that $\widehat{\rho}_g=0$ and $\widehat{\rho}_{s,i}=0$ for $i\neq i^*$. Therefore, we can now rewrite Equation (7) as

$$\prod_{i=1}^{q}\mathsf{S}_i^2\widehat{\rho}_{s,i^*}\mathsf{S}_{i^*}^{-1}\cdot\left(\zeta_{z,i^*}\mathsf{Z}^{(i^*)}+\sum_{i=1}^{q}\left(\sum_{j'=1}^{m}\sum_{j=1}^{\ell}\zeta_{t,i,j',j}\mathsf{T}_{j',j}^{(i)}+\sum_{j=1}^{n}\zeta_{\bar{t},i,j}\overline{\mathsf{T}}_j^{(i)}\right)\right)=\prod_{i=1}^{q}\mathsf{S}_i^2\left(1+\sum_{k=1}^{n}\mathsf{X}_k(r_k^*+\sum_{j=1}^{m}\vec{\mu}_{k,j}^*\mathsf{H}_j)\right).$$

Now, notice that in the RHS there are no monomials with $\mathsf{S}_{i^*}^{-1}\prod_{i=1}^{q}\mathsf{S}_i^2$, meaning for $i \neq i^*$,

$$\sum_{j'=1}^{m}\sum_{j=1}^{\ell}\zeta_{t,i,j',j}\mathsf{T}_{j',j}^{(i)} + \sum_{j=1}^{n}\zeta_{\bar{t},i,j}\overline{\mathsf{T}}_{j}^{(i)} = 0\,.$$

Hence, we can conclude that

$$\prod_{i=1}^{q}\mathsf{S}_i^2\widehat{\rho}_{s,i^*}\mathsf{S}_{i^*}^{-1}\cdot\left(\zeta_{z,i^*}\mathsf{Z}^{(i^*)} + \sum_{j'=1}^{m}\sum_{j=1}^{\ell}\zeta_{t,i^*,j',j}\mathsf{T}_{j',j}^{(i^*)} + \sum_{j=1}^{n}\zeta_{\bar{t},i^*,j}\overline{\mathsf{T}}_{j}^{(i^*)}\right) = \prod_{i=1}^{q}\mathsf{S}_i^2\left(1+\sum_{k=1}^{n}\mathsf{X}_k(r_k^* + \sum_{j=1}^{m}\vec{\mu}_{k,j}^*\mathsf{H}_j)\right)\,.$$

Expanding the LHS with how $\mathsf{X},\mathsf{T},\overline{\mathsf{T}}$ are defined, we have that

$$\text{LHS} = \prod_{i=1}^{q}\mathsf{S}_i^2\widehat{\rho}_{s,i^*}\left(\zeta_{z,i^*} + \sum_{k=1}^{n}\mathsf{X}_k\left(\zeta_{z,i^*}(r_k^{(i^*)} + \sum_{j'=1}^{m}\mu_{k,j'}^{(i^*)}\mathsf{H}_{j'}) + \sum_{j'=1}^{m}\sum_{j=1}^{\ell}\zeta_{t,i^*,j',j}A_{j,k}\mathsf{H}_{j'} + \zeta_{\bar{t},i^*,k}\right)\right)$$

Since $\widehat{\rho}_{s,i^*}\zeta_{z,i^*} = 1$, by equating the coefficient with the RHS on the monomials $1, \mathsf{X}_k, \mathsf{X}_k\mathsf{H}$, we have the given equations in the lemma statement; thus, proving the lemma. □

With Lemma 12, we now have that the adversary does not win in the unforgeability game. Thus, Bad_1 cannot occur due to a contradiction.

ANALYSIS OF EVENT Bad_2. Note first that since the view of $\mathcal{A}$ is indistinguishable from its view in the unforgeability game, the values of $\vec{a},\vec{a}',\vec{r}$ are information theoretically hidden from $\mathcal{A}$. Now, we consider when Bad_2 occurs. In this case, we have that $\mathsf{V}_1 \neq 0$ or $\mathsf{V}_2 \neq 0$ but $\mathsf{Q}_1(\mathsf{Y}) = 0$ or $\mathsf{Q}_2(\mathsf{Y}) = 0$. We want to bound the probability that $\mathsf{Q}_1(\mathsf{Y}) = 0$ or $\mathsf{Q}_2(\mathsf{Y}) = 0$ given that $\mathsf{V}_1 \neq 0$ or $\mathsf{V}_2 \neq 0$ and $\mathcal{A}$ wins the game.

To do so, assume w.l.o.g., that $\mathsf{V}_2 \neq 0$, then we will show that $\mathsf{Q}_2 = 0$ with probability at most $(2q+3)/p$. First, notice that $(\vec{r},\vec{r}\circ\vec{c})$ where $\vec{r}\leftarrow\!\!\$\,(\mathbb{Z}_p^*)^q, \vec{c}\leftarrow\!\!\$\,\mathbb{Z}_p$ and $(\vec{r},\vec{c'})\leftarrow\!\!\$\,(\mathbb{Z}_p^*)^q\times\mathbb{Z}_p^q$ are distributed identically. Next, let $\mathsf{V}_2'(\vec{\mathsf{A}},\vec{\mathsf{B}},\vec{\mathsf{A}}',\vec{\mathsf{B}}',\vec{\mathsf{R}}',\mathsf{C},\mathsf{Y})$ be defined as

$$\mathsf{V}_2'(\vec{\mathsf{A}},\vec{\mathsf{B}},\vec{\mathsf{A}}',\vec{\mathsf{B}}',\vec{\mathsf{R}}',\mathsf{C},\mathsf{Y}) := \mathsf{V}_2(\vec{\mathsf{A}}\mathsf{Y}+\vec{\mathsf{B}},\vec{\mathsf{A}}'\mathsf{Y}+\vec{\mathsf{B}}',\vec{\mathsf{R}}'\mathsf{Y}+\vec{\mathsf{C}}')\,.$$

Now, we will employ the following lemma from [BFL20, Lemma 2.1].

Lemma 13 (Lemma 2.1 of [BFL20])**.** Let P be a non-zero multivariate polynomial in $\mathbb{Z}_p[\mathsf{X}_1,\ldots,\mathsf{X}_n]$ of total degree d. Let $\mathsf{Q}(\mathsf{Z}) \in (\mathbb{Z}_p[\mathsf{Y}_1,\ldots,\mathsf{Y}_n,\mathsf{Y}_1',\ldots,\mathsf{Y}_n])[\mathsf{Z}]$ be such that $\mathsf{Q}(\mathsf{Z}) := \mathsf{P}(\mathsf{Y}_1\mathsf{Z}+\mathsf{Y}_1',\ldots,\mathsf{Y}_n\mathsf{Z}+\mathsf{Y}_n')$. Then, the coefficient of the maximal degree of Q is a polynomial in $\mathbb{Z}_p[\mathsf{Y}_1,\ldots,\mathsf{Y}_n]$ of degree d.

Thus, the leading coefficient in the variable Y of V_2' is some non-zero polynomial $\mathsf{V}_{2,\max}'(\vec{\mathsf{A}},\vec{\mathsf{A}}',\vec{\mathsf{R}})$. Since $\vec{a},\vec{a}',\vec{r}$ are information theoretically hidden from $\mathcal{A}$ and they are uniformly random in $\mathbb{Z}_p^*$, we have that $\mathsf{Pr}[\mathsf{V}_{2,\max}'(\vec{a},\vec{a}',\vec{r}) = 0|\neg\mathsf{Bad}_1 \wedge (\mathcal{A}\text{ wins UNF game})] \leqslant \frac{2q+3}{p}$ by Schwartz-Zippel lemma and that the degree of V_2 is at most $2q+3$.

Thus,

$$\begin{aligned}\mathsf{Pr}[\mathsf{Bad}_2 \wedge (\mathcal{A}\text{ wins UNF game})] &\leqslant \mathsf{Pr}[\mathsf{V}_1 \neq 0 \vee \mathsf{V}_2 \neq 0|\neg\mathsf{Bad}_1 \wedge (\mathcal{A}\text{ wins UNF game})]\\ &\leqslant \mathsf{Pr}[\mathsf{V}_{2,\max}'(\vec{a},\vec{a}',\vec{r}) = 0|\neg\mathsf{Bad}_1 \wedge (\mathcal{A}\text{ wins UNF game})] \leqslant \frac{2q+3}{p}\,.\end{aligned}$$

Hence, we can conclude that

$$\mathsf{Adv}_{\mathsf{SEQ}}^{\mathsf{unf}}(\mathcal{A},\lambda) \leqslant \mathsf{Adv}_{\mathsf{GGen}}^{q\text{-powdendl}}(\mathcal{B},\lambda) + \frac{3q+3}{p}\,.$$

□

C Security Proofs of SEQ-Based SSA

In this section, we give the proof of Theorem 4.

CORRECTNESS. Correctness follows from correctness and adaption of SEQ.

UNFORGEABILITY. We state the following lemma with concrete security bound reducing the unforgeability of $\mathsf{SSA}_{\mathsf{SEQ}}$ to unforgeability of $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, A]$ where is the corresponding full-row-rank linear map defining the equivalence class.

Lemma 14. Let GGen be a bilinear group parameter generator outputting groups of prime-order $p = p(\lambda)$, $A = [\vec{0} \| \vec{1}\| - \mathbb{I}_{n-1}] \in \mathbb{Z}_p^{(n-1)\times(n+1)}$, and $\mathsf{SSA}_{\mathsf{SEQ}} = \mathsf{SSA}_{\mathsf{SEQ}}[\mathsf{GGen}, m, n], \mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, m, n+1, A]$. Then, for any adversary $\mathcal{A}$ (making at most Q_{Iss} issuance query and running in time $t_{\mathcal{A}}$), there exists an adversary $\mathcal{B}$ running in time roughly that of $\mathcal{A}$ and making at most Q_{Iss} signing queries

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SSA}_{\mathsf{SEQ}},n}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SEQ},n}(\mathcal{B}, \lambda) \,.$$

Proof. Consider an adversary $\mathcal{A}$ winning in the unforgeability game of $\mathsf{SSA}_{\mathsf{SEQ}}$. In particular, it does the following: (1) makes issuance queries of the form $\phi_{\mathsf{info}_i, \vec{v}_i}$ with $(\mathsf{info}_i \in \mathbb{Z}_p, \vec{v}_i \in \mathbb{Z}_p^m))$ and receives a signature $\sigma_{\mathsf{SEQ},i}$ of $\vec{\mu} = (\vec{v}, \vec{0} \in \mathbb{Z}_p^{m(n-1)}, (\mathsf{info}, \vec{0} \in \mathbb{Z}_p^{m-1}))$ and (2) returns a forgery $(\tau^* = ((\widetilde{C}_i^*)_{i\in[n+1]}, \sigma^*, r^*_{n+1}), (\vec{s}_i^* \in \mathbb{Z}_p^m, \tau_i^* = r_i^*)_{i\in[n]})$ for predicate $\phi^* = \phi_{\mathsf{info}^*}$ such that info^* was not queried before, or $\vec{v}^* = \sum_{i=1}^n \vec{s}_i^*$ is not one that was sent during issuance queries. In this case, we can construct a trivial reduction $\mathcal{B}$ which (a) simulates each issuance query with a signing query on $\vec{\mu} = (\vec{v}_i, \vec{0}, (\mathsf{info}_i, \vec{0}))$ with randomness $\vec{0}$ and (b) on the forgery, forwards to its game a forgery of the form $(\vec{\mu}^* = (\vec{s}_1, \ldots, \vec{s}_n, (\mathsf{info}, \vec{0})), \vec{r}^* = (r_1^*, \ldots, r_n^*, r_{n+1}^*), \sigma^*)$. Correctness and efficiency of the reduction follows immediately, since $\vec{\mu}^*$ is not in any of the signed equivalence classes as discussed above. □

UNLINKABILITY. The following lemma states the concrete security guarantee for unlinkability of $\mathsf{SSA}_{\mathsf{SEQ}}$ construction. Note that with $\mathsf{Adv}^{\mathsf{adapt}}_{\mathsf{SEQ}}(\mathcal{B}, \lambda) = 0$, we have that unlinkability of $\mathsf{SSA}_{\mathsf{SEQ}}$ is perfect.

Lemma 15. Let GGen be a bilinear group parameter generator outputting groups of prime-order $p = p(\lambda)$, $A = [\vec{0} \| \vec{1}\| - \mathbb{I}_{n-1}] \in \mathbb{Z}_p^{(n-1)\times(n+1)}$, and $\mathsf{SSA}_{\mathsf{SEQ}} = \mathsf{SSA}_{\mathsf{SEQ}}[\mathsf{GGen}, m, n], \mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, m, n, A]$. Then, for any adversary $\mathcal{A}$ (making Q_{Share} queries to Share), we have that there exists an adversary $\mathcal{B}_1, \mathcal{B}_2$ against the adaption property of SEQ such that

$$\mathsf{Adv}^{\mathsf{unlk}}_{\mathsf{SSA}_{\mathsf{SEQ}},n}(\mathcal{A}, \lambda) \leqslant 2Q_{\mathrm{Share}} \left(\mathsf{Adv}^{\mathsf{adapt}}_{\mathsf{SEQ}}(\mathcal{B}_1, \lambda) + n \cdot \mathsf{Adv}^{\mathrm{hid}}_{\mathsf{SEQ}}(\mathcal{B}_2, \lambda) \right) \,.$$

Proof. Consider an adversary $\mathcal{A}$ playing the unlinkability game, picking a possibly malicious public key pk, invoking user-side oracles $\mathrm{U}_1, \mathrm{U}_2$ for $\beta = 0, 1$ (possibly issuing maliciously formed credentials), and querying the share oracle Share at most Q_{Share} times. Note that in U_2, the game checks whether the issued credential is valid or not, and in Share, the adversary only learns $n-1$ out of n shares. Also, by the condition in Share that the specified predicate ϕ should be satisfied by both inputs $v_0' = (\mathsf{info}_0, \vec{v}_0 \in \mathbb{Z}_p^m), v_1' = (\mathsf{info}_1, \vec{v}_1 \in \mathbb{Z}_p^m)$, we assume w.l.o.g. that public tags are equal $\mathsf{info}_0 = \mathsf{info}_1$ (otherwise, $\mathcal{A}$ lose in the game). Now, we consider the following sequence of games $\mathbf{G}_{i,b}$ parameterized by an index $i \in [0, 3]$ and a bit b–the bit denotes which bit b is used in the unlinkability game.

- $\mathbf{G}_{0,b}$: This is exactly the game $\mathrm{UNLK}_{\mathsf{SSA}_{\mathsf{SEQ}},n,b}$.
- $\mathbf{G}_{1,b}$: In this game, for the oracle Share with input β, we sample $\vec{s} \in (\mathbb{Z}_p^m)^n, \vec{r} \in \mathbb{Z}_p^n, (\widetilde{C}_i)_{i\in[n+1]}$, and $\widetilde{\sigma}_{\mathsf{SEQ}}$ by (1) randomly sample $\vec{s} \in (\mathbb{Z}_p^m)^n$ as additive secret share of $\vec{v}_\beta$ (or $\vec{v}_{1-\beta}$ depending on which on the bit b of the game uses), (2) sample $\vec{r}$ uniformly at random, (3) compute $(\widetilde{C}_i)_{i\in[n+1]} \leftarrow \mathsf{SEQ.Com}(\mathsf{pp}, (\vec{s}, (\mathsf{info}, \vec{0})); \vec{r})$, and (4) (inefficiently) sample $\widetilde{\sigma}_{\mathsf{SEQ}}$ randomly from the set of valid signatures with respect to $\mathsf{SEQ.VerAdapt}$ on $(\widetilde{C}_i)_{i\in[n+1]}$.

  This game is indistinguishable from $\mathbf{G}_{0,b}$ due to the adaption property of SEQ, and in particular, via a simple hybrid argument (sequentially changing how each query to Share is answered)

$$|\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{0,b}(\lambda) = 1] - \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{1,b}(\lambda) = 1]| \leqslant Q_{\mathrm{Share}} \cdot \mathsf{Adv}^{\mathsf{adapt}}_{\mathsf{SEQ}}(\mathcal{B}_1, \lambda) \,.$$

- $\mathbf{G}_{2,b}$: On each query to Share with the subset $S \subsetneq [n]$, for each $i \in [n] \backslash S$ (i.e., the non-opened indices) compute $\widetilde{C}_i \leftarrow \mathsf{SEQ.Com}(\mathsf{pp}, \vec{0}; r_i)$ for $r_i \leftarrow\!\!\$\, \mathbb{Z}_p$ instead. Since the oracle does not open $\widetilde{C}_{i*}$, this is indistinguishable by hiding property of SEQ, meaning via a simple hybrid argument (over at most nQ_{Share} commitments which we apply the property to), there exists an adversary $\mathcal{B}_2$ such that

$$|\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{1,b}(\lambda) = 1] - \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{2,b}(\lambda) = 1]| \leqslant Q_{\mathrm{Share}} n \cdot \mathsf{Adv}^{\mathrm{hid}}_{\mathsf{SEQ}}(\mathcal{B}_2, \lambda) \,.$$

- $\mathbf{G}_{3,b}$: Now, in each Share oracle, instead of computing $\vec{s}_i$ as secret shares of $\vec{v}_0$ or $\vec{v}_1$, we compute them as secret shares of $\vec{0}$. Since only $n-1$ of these shares are revealed to the adversary, no information about the secret is revealed to the adversary (i.e., perfect privacy of the secret sharing scheme), this game is indistinguishable from $\mathbf{G}_{2,b}$. Hence, $\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{2,b}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_{3,b}(\lambda) = 1]$.

Finally, we can see that $\mathbf{G}_{3,0} = \mathbf{G}_{3,1}$ since Share does not rely on $\vec{v}_0$ nor $\vec{v}_1$. The bound in the lemma statement follows from the game hops. □

D Threshold Secret Sharing Extensions

In this section, we describe how to extend our SSA schemes to accommodate threshold access structure via Shamir's secret sharing.

D.1 BBS-based SSA for Shamir Secret Sharing

We now detail the necessary changes to $\mathsf{SSA}_{\mathsf{BBS}}$ from Section 5 in order to incorporate $(t+1)$-out-of-n Shamir secret sharing scheme (t being the maximum number of shares of which the secret is still hidden) instead. We denote the modified scheme $\mathsf{TSSA}_{\mathsf{BBS}} = \mathsf{TSSA}_{\mathsf{BBS}}[\mathsf{GGen}, m, n]$ The changes are very minimal and applied to only the sharing algorithm and the recovery algorithm, as detailed below:

Sharing Algorithm. We make the following changes.

- The shares $\vec{s}_1, \ldots, \vec{s}_n \in \mathbb{Z}_p^m$ are now computed as Shamir secret share of $\vec{v}$. More precisely, for randomness $\vec{\rho}_1, \ldots, \vec{\rho}_t \leftarrow\!\!\$\, \mathbb{Z}_p^m$, the shares are defined as

$$\vec{s}_i = \vec{v} + \sum_{j=1}^{t} i^j \cdot \vec{\rho}_j \,.$$

- The proof system Π_{Share} now proves knowledge of witnesses in the following relation.

$$\mathsf{R}_{\mathsf{Share}} := \left\{ \begin{array}{c} ((G, \vec{H}, \widetilde{A}, \widetilde{B}, (C_i)_{i\in[n]}, \mathsf{info}); \\ (\alpha, e, \vec{v} \in \mathbb{Z}_p^m, \vec{s} \in (\mathbb{Z}_p^m)^n, \vec{\rho} \in (\mathbb{Z}_p^m)^t, \vec{r} \in \mathbb{Z}_p^n)) \end{array} : \begin{array}{c} \widetilde{B} = \alpha(G + \sum_{j=1}^m v_j H_j + \mathsf{info} H_{n+1}) - e\widetilde{A} \wedge \\ \Big(\forall i \in [n] : \vec{s}_i = \vec{v} + \sum_{j=1}^t i^j \cdot \vec{\rho}_j \\ \wedge\, C_i = r_i G + \sum_{j=1}^m s_{i,j} H_j\Big) \end{array} \right\} .$$

  Note that one can implement this with the proof system Π_{Lin} from Appendix A.2 for the linear relation with the

following matrix M_{Share} (omitted cells are $0_{\mathbb{G}}$), group elements $\vec{Y}_{\mathsf{Share}}$ and witness $\vec{x}_{\mathsf{Share}} \in \mathbb{Z}_p^{n(m+1)+2}$:

$$M_{\mathsf{Share}} := \begin{pmatrix} G + \mathsf{info} H_{m+1} & -\widetilde{A} & H_1 & \cdots & H_m & & & & & & \\ -C_1 & & H_1 & \dots & H_m & \dots & 1^t H_1 & \dots & 1^t H_m & G & \\ \vdots & & & \vdots & & \ddots & & \vdots & & & \ddots \\ -C_n & & H_1 & \dots & H_m & \dots & n^t H_1 & \dots & n^t H_m & & G \end{pmatrix},$$

$$\vec{Y}_{\mathsf{Share}} := \begin{pmatrix} \widetilde{B} \\ \vec{0}_{\mathbb{G}} \end{pmatrix}, \vec{x}_{\mathsf{Share}} := \begin{pmatrix} \alpha \\ e \\ \alpha\vec{v} \in \mathbb{Z}_p^m \\ \alpha\vec{\rho}_1 \in \mathbb{Z}_p^m \\ \vdots \\ \alpha\vec{\rho}_t \in \mathbb{Z}_p^m \\ \alpha\vec{r} \in \mathbb{Z}_p^n \end{pmatrix}.$$

Similarly to the discussion in Appendix A.2, the knowledge-soundness extractor will extract a witness of the following relaxed relation:

$$\widetilde{\mathsf{R}}_{\mathsf{Share}} := \left\{ \begin{array}{rcl} (G, \vec{H}, \widetilde{A}, \widetilde{B}, (C_i)_{i\in[n]}, \mathsf{info}); & : & \widetilde{B} = \alpha(G + \mathsf{info} H_{m+1}) + \sum_{j=1}^m v'_j H_j - e\widetilde{A} \wedge \\ (\alpha, e, \vec{v}', \vec{\rho}', \vec{r}') & & \forall i \in [n] : \alpha C_i = r'_i G + \sum_{j=1}^m \left(v'_j + \sum_{k=1}^t i^k \rho'_{k,j} \right) H_j \end{array} \right\}.$$

There is no guarantee that $\alpha \neq 0$, but analogously, this implies breaking the SUF-BBS security.

Recovery Algorithm. The recovery algorithm now recovers the secret via the recovery algorithm of Shamir secret sharing, i.e., given $t+1$ shares $(i, \vec{s}_i)$ interpolate a polynomial f with coefficient over $\mathbb{Z}_p^m$ such that $f(i) = \vec{s}_i$, and outputting the constant term as $\vec{v}$.

Theorem 16. The scheme $\mathsf{TSSA}_{\mathsf{BBS}} = \mathsf{TSSA}_{\mathsf{BBS}}[\mathsf{GGen}, m, n]$ satisfies correctness, unforgeablility, unlinkablility, and blind issuance.

Proof sketch. Most of the proofs follows from the security proofs of $\mathsf{SSA}_{\mathsf{BBS}}$. In particular, correctness and blind issuance are trivially satisfied. We mention the distinctions for unforgeability and unlinkability below.

UNFORGEABILITY. We now consider an adversary winning in the unforgeability game and our goal is to construct a reduction $\mathcal{B}$ against augmented unforgeability of BBS. The reduction differs from the one from Appendix A.3 as follows:

- $\mathbf{G}_1$: The extracted witness is now of the form $(\alpha, e, \vec{v}', \vec{\rho}' \in (\mathbb{Z}_p^m)^t, \vec{r}')$ for $\widetilde{\mathsf{R}}_{\mathsf{Share}}$ defined in this section.
- $\mathbf{G}_2$: The game aborts if $\alpha \neq 0$ but for some $i \in [n]$, the shares $\vec{s}_i^* \neq \alpha^{-1}(\vec{v}' + \sum_{j=1}^t i^j \vec{\rho}'_i)$ (i.e., reconstructing the shares from). This breaks binding of the commitment C_i. Hence, the abort still implies breaking the discrete logarithm assumption. If this abort does not occur, we know that running $\mathsf{Recover}$ on $(\vec{s}_i^*)_{i\in[n]}$ would result in $\alpha^{-1}\vec{v}'$.
- Extracting SUF-BBS forgery: Here, we still have the two cases (a) $\alpha = 0$ or (b) $\alpha \neq 0$ and the abort introduced in game $\mathbf{G}_2$ does not occur. For (a), we get the special forgery for SUF-BBS. For (b), since the abort does not occur, we have that the algorithm $\mathsf{Recover}$ returns $\alpha^{-1}\vec{v}'$. Hence, an adversary winning in $\mathbf{G}_2$ breaks SUF-BBS security of BBS signatures.

The concrete security bounds are the same.

UNLINKABILITY. Unlinkability proof follows similar sequence of games as the proof of Lemma 8 with the following changes:

- The games $\mathbf{G}_{1,b}$ and $\mathbf{G}_{2,b}$ are unchanged.

- $\mathbf{G}_{3,b}$: We now consider Share queries with subset $S \subsetneq [n]$ of size $|S| \leqslant t$. Let $S' = [n] \backslash S$. Then, this game now computes $\widetilde{C}_i \leftarrow r_i \mathbb{G}$ with $r_i \leftarrow\!\!\$\, \mathbb{Z}_p$ for $i \in S'$. By perfectly hiding of Pedersen commitment, this game is identical to the previous game.
- $\mathbf{G}_{4,b}$: For each Share oracle query, instead of computing $\vec{s}_i$ as Shamir's secret shares of $\vec{v}_0$ or $\vec{v}_1$, we compute them as Shamir's secret shares of $\vec{0}$. Since at most t of these shares are revealed, the view of the adversary remains identical to that of $\mathbf{G}_{3,b}$. Also, $\mathbf{G}_{4,0} = \mathbf{G}_{4,1}$ since Share does not rely on $\vec{v}_0$ nor $\vec{v}_1$.

□

D.2 SEQ-based SSA for Shamir Secret Sharing

In this section, we describe the construction for SEQ-based SSA scheme where the equivalence class is over Shamir secret sharing of a secret $\vec{v} \in \mathbb{Z}_p^m$. In particular, observe that the sharing algorithm for $(t+1)$-out-of-n Shamir secret sharing can be viewed as the following linear map:

$$A = \begin{pmatrix} 1 & \dots & 1^t \\ \vdots & \ddots & \vdots \\ 1 & \dots & n^t \end{pmatrix} \in \mathbb{Z}_p^{n \times (t+1)}$$

In particular, the secret shares $\vec{s} = (\vec{s}_1^T, \dots, \vec{s}_n^T)^T \in (\mathbb{Z}_p^m)^n$ are such that $(A \otimes \mathbb{I}_m)(\vec{v}^T, \vec{r}^T)^T = \vec{s}$, where $A \otimes B$ denotes the Kronecker/tensor product and $\vec{r} = (\vec{r}_1, \dots, \vec{r}_t) \in (\mathbb{Z}_p^m)^t$ is the randomness of the secret sharing scheme, i.e., the coefficients of a degree-t $\mathbb{Z}_p$-polynomial. Note that the matrix $\overline{A}$ which is the transpose of the matrix truncating the first column of A is exactly the matrix which defines the equivalence class. In particular, we have

$$\overline{A} = \begin{pmatrix} 1 & \dots & n \\ \vdots & \ddots & \vdots \\ 1 & \dots & n^t \end{pmatrix} \in \mathbb{Z}_p^{t \times n}, \tag{8}$$

which is a full-row-rank matrix. The equivalence class is defined with respect to $\overline{A}$ as in Equation (3). Note in particular that signing an equivalence class for a value v is done by signing the vector $\vec{\mu} = (\vec{v}, \dots, \vec{v}) \in (\mathbb{Z}_p^m)^n$, i.e., Shamir secret share of $\vec{v}$ with randomness $\vec{r} = \vec{0}$.

Now, with the sketched observation above, we described the particular modifications made to $\mathsf{SSA}_{\mathsf{SEQ}}$ into $\mathsf{TSSA}_{\mathsf{SEQ}}$ which supports threshold secret sharing.

Setup and Key generation. These remain identical to that of $\mathsf{SSA}_{\mathsf{SEQ}}$; however, we note that the matrix A which is the parameter of the $\mathsf{SEQ} = \mathsf{SEQ}[\mathsf{GGen}, m, n, A]$ scheme is now $\overline{A}$ from Equation (8).

Issuance: As mentioned above, the message committed to and signed by SEQ's secret key is $\vec{\mu} = (\vec{v}, \dots, \vec{v}) \in (\mathbb{Z}_p^m)^n$. The user's side remains unchanged.

Sharing: The adaption randomness space $\mathcal{R}^{\mathsf{Adapt}}$ is now changed to $(\mathbb{Z}_p^m)^t \times \mathbb{Z}_p^n$. However, the

Verification: The verification algorithms VerPub and $\mathsf{VerShare}$ are unchanged.

Recovery algorithm: The recovery algorithm on at least $t+1$ shares of the form $\mathsf{ss}_i = (\mathsf{info}, \vec{s}_i)$ recovers the secret by interpolating a polynomial f of degree at most t with coefficient over $\mathbb{Z}_p^m$ such that $f(i) = \vec{s}_i$, and outputting the constant term as $\vec{v}$. If no such f exists, return $\perp$.

Theorem 17. The scheme $\mathsf{TSSA}_{\mathsf{SEQ}} = \mathsf{TSSA}_{\mathsf{SEQ}}[\mathsf{GGen}, m, n]$ satisfies correctness, unforgeablility, unlinkablility, and blind issuance.

Proof sketch. Most of the proofs follows from the security proofs of $\mathsf{SSA}_{\mathsf{SEQ}}$. In particular, correctness and blind issuance are trivially satisfied. We mention the distinctions for unforgeability and unlinkability below.

UNFORGEABILITY. We now consider an adversary winning in the unforgeability game and our goal is to construct a reduction $\mathcal{B}$ against unforgeability of $\mathsf{SEQ}[\mathsf{GGen}, m, n, A]$. The reduction differs from the one from Appendix C as follows:

- Each issuance query (indexed by $i \in [Q_{\mathrm{Iss}}]$) $\mathsf{info}_i, \vec{v}_i \in \mathbb{Z}_p^m$ is turned into a signing query of the form $\vec{\mu}_i = (\vec{v}_i, \ldots, \vec{v}_i, (\mathsf{info}_i, \vec{0} \in \mathbb{Z}_p^{m-1}))$. The resulting signature is forwarded to the adversary.
- The valid forgery $(\tau^* = ((\widetilde{C}_i^*)_{i\in[n+1]}, \sigma^*, r_{n+1}^*), (\vec{s}_i^* \in \mathbb{Z}_p^m, \tau_i^* = r_i^*)_{i\in[n]})$ for the predicate $\phi_{\mathsf{info}*}$ is such that either (1) $\mathsf{Recover}(\mathsf{pp}, (\vec{s}_i^*)_{i\in[n]})$ is $\perp$ (not valid Shamir secret shares) or does not correspond to any $\vec{v}_i$ or (2) info^* does not correspond to any info_i.

For the forgery, both cases imply that the message $\vec{\mu}^* = (\vec{s}_1^*, \ldots, \vec{s}_n^*, (\mathsf{info}^*, \vec{0}))$ is not in any of the equivalence classes $[\![\vec{\mu}_i]\!]$. Therefore, combined with the randomness $\vec{r}^* = (r_1^*, \ldots, r_{n+1}^*)$ and that σ^* is valid when verified with $(\widetilde{C}_i^*)_{i\in[n+1]}$ which opens to $\vec{\mu}^*$ and $\vec{r}^*$, we have that $(\vec{\mu}^*, \vec{r}^*, \sigma^*)$ is a valid forgery of SEQ.

UNLINKABILITY. Unlinkability proof follows similar sequence of games as the proof of Lemma 15 with the following changes:

- $\mathbf{G}_{1,b}$ is unchanged.
- $\mathbf{G}_{2,b}$: We now consider Share queries with subset $S \subsetneq [n]$ of size $|S| \leqslant t$. Let $S' = [n] \backslash S$. Then, this game now computes $\widetilde{C}_i \leftarrow \mathsf{SEQ.Com}(\mathsf{pp}, \vec{0}; r_i)$ with $r_i \leftarrow\!\!\$\, \mathbb{Z}_p$ for $i \in S'$. By standard hybrid argument, we have that

$$|\mathsf{Pr}[\mathbf{G}_{1,b}^{\mathcal{A}}(\lambda) = 1] - \mathsf{Pr}[\mathbf{G}_{2,b}^{\mathcal{A}}(\lambda) = 1]| \leqslant Q_{\mathrm{Share}} n \cdot \mathsf{Adv}_{\mathsf{SEQ}}^{\mathrm{hid}}(\mathcal{B}_2, \lambda) \,.$$

- $\mathbf{G}_{3,b}$: For each Share oracle query, instead of computing $\vec{s}_i$ as Shamir's secret shares of $\vec{v}_0$ or $\vec{v}_1$, we compute them as Shamir's secret shares of $\vec{0}$. Since at most t of these shares are revealed, the view of the adversary remains identical to that of $\mathbf{G}_{2,b}$. Also, $\mathbf{G}_{3,0} = \mathbf{G}_{3,1}$ since Share does not rely on $\vec{v}_0$ nor $\vec{v}_1$.

□

Remark 18 (Extension to any linear secret sharing schemes.)**.** We can also extend this idea to any linear secret sharing scheme where the sharing matrix $A \in \mathbb{Z}_p^{n\times\ell}$ which has full-column-rank. (An this is to be expected, otherwise there is a redundancy in the randomness for the secret sharing scheme.) Signing an equivalence class of a value v is done by signing $A^T(v, \vec{0})^T$.