Everlasting Anonymous Rate-Limited Tokens*

Rutchathon Chairattana-Apirom[1], Nico Döttling[2], Anna Lysyanskaya[3], and Stefano Tessaro[1]

[1] Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, US
{rchairat,tessaro}@cs.washington.edu
[2] CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
doetting@cispa.de
[3] Brown University, Providence, RI, US
anna_lysyanskaya@brown.edu

Abstract. *Anonymous rate-limited tokens* are a special type of credential that can be used to improve the efficiency of privacy-preserving authentication systems like Privacy Pass. In such a scheme, a user obtains a "token dispenser" by interacting with an issuer, and the dispenser allows the user to create up to a pre-determined number k of unlinkable and publicly verifiable tokens. Unlinkable means that one should not be able to tell that two tokens originate from the same dispenser, but also they cannot be linked to the interaction that generated the dispenser. Furthermore, we can limit the rate at which these tokens are created by linking each token to a context (e.g., the service we are authenticating to), and imposing a limit $N \leqslant k$ such that seeing more than N tokens for the same context will reveal the identity of the user. Constructions of such tokens were first given by Camenisch, Hohenberger and Lysyanskaya (EUROCRYPT '05) and Camenisch, Hohenberger, Kohlweiss, Lysyanskaya, and Meyerovich (CCS '06).

In this work, we present the first construction of *everlasting* anonymous rate-limited tokens, for which unlinkability holds against computationally *unbounded* adversaries, whereas other security properties (e.g., unforgeability) remain computational. Our construction relies on pairings. While several parameters in our construction unavoidably grow with k, the key challenge we resolve is ensuring that the complexity of dispensing a token is *independent* of the parameter k.

We are motivated here by the goal of providing solutions that are robust to potential future quantum attacks against the anonymity of previously stored tokens. A construction based on post-quantum secure assumptions (e.g., based on lattices) would be rather inefficient—instead, we take a pragmatic approach dispensing with post-quantum security for properties not related to privacy.

1 Introduction

This paper is concerned with the problem of building anonymous credentials, and more precisely rate-limited tokens, for which anonymity is *everlasting*, i.e., it holds *unconditionally*, whereas other properties hold computationally. Our main goal is to benefit from existing group-theoretic machinery for efficiency, while at the same time resisting store-now/break-later attacks against anonymity by future quantum computers.

The key feature of anonymous credentials [Cha82, CL01, CL03, CL04] is that they are issued via a secure protocol between an issuer and a user, where the issuer need not know or learn the full set of attributes it is certifying; furthermore, they are demonstrated anonymously by the user via a zero-knowledge proof, so that the verifier need not learn anything about the user other than the fact that the user's identity attributes satisfy a given access policy. They are the preferred solution for ongoing efforts to develop digital identity apps [ARF24], as they enforce data minimization and ensure that citizens do not leave behind a trace that would allow linking different transactions of the same user.

In many contexts, it is important to limit the number of times a credential is demonstrated by a particular user. For example, a credential may warrant access to a service only for a bounded number of times, such

* A preliminary version of this paper appears in the proceedings of ASIACRYPT 2025. This is the full version.

as in web authentication tokens like Privacy Pass [DGS+18]. (In fact, Privacy Pass itself relies on one-time tokens that can only be used once.) A similar scenario arises in e-cash tokens, which can be spent within some limited value. Another common reason to limit the number of demonstrations of a credential is that we want to limit the damage arising from an adversary who has obtained a credential.

Anonymous Credentials, E-Cash and E-Tokens with Context-Based Restrictions. One way to mitigate the extent to which a rogue user can overshare an anonymous credential is to require that there be a unique, deterministic pseudonym that corresponds to this user's relationship to each verifier it interacts with, or, more generally, for any given transaction's context. That way, the user cannot create multiple Sybils for the same credential. This was part of early work on DAA [BCC04]: this pseudonym was computed as a pseudorandom function of the user's secret key (which is one of the attributes of its anonymous credential) and the verifier's identity. Thus, there was a *context-based* limit: for each verifier, the user could establish just a unique pseudonym.

Camenisch, Hohenberger and Lysyanskaya [CHL06] (CHL) extended this idea to develop an e-cash system that incorporated context—i.e., the merchant's identity—into the e-coin to achieve money-laundering protection. The context-based limit here is that a user could not anonymously spend more than N e-coins with the same merchant. The main idea behind the CHL approach is to also think of an e-coin as a credential where one of the attributes is a seed/key s for a pseudorandom function $F_s(\cdot)$. Each transaction includes a serial number sn computed as $\mathsf{sn} = F_s(\mathsf{ctxt}, i)$ and a proof that sn was computed correctly for some $0 \leqslant i < N$. This ensures that at most N e-coins can be spent in any given context ctxt; for example, if the context is the identity of the merchant, that limits the total number of coins that can be spent with this merchant. (An additional serial number, computed without the context, is also included to enforce a total spending limit.) Following on their earlier work on compact ecash [CHL05], the CHL solution also provides a mechanism to recover the user's identity in the event that the same serial number sn was used in more than one transaction.

Camenisch, Hohenberger, Kohlweiss, Lysyanskaya and Meyerovich [CHK+06] (CH-KLM) later extended this approach to compact e-tokens. Here, instead of a total limit on the number of transactions with a particular credential, or a limit per merchant, it was the rate of transactions that was limited, for example up to N per day. To accommodate this use case, the serial number is still $\mathsf{sn} = F_s(\mathsf{ctxt}, i)$, but now the context ctxt is the date.

Towards post-quantum "privacy". Crucially, the CHL/CHKLM architecture is generic and can be instantiated with any signature scheme, PRF, and zero-knowledge proof system; so, should our goal be to achieve security in the post-quantum setting, all instantiated components would need to be post-quantum secure primitives—the resulting construction would however be fairly expensive.

However, post-quantum-unforgeable digital signatures can be an overkill in this scenario, as non-post-quantum signatures would be forgeable only if a quantum computer is indeed available. Until then, a more pragmatic and efficient approach may be preferable, namely one where:

- We stick to group-based signatures, such as CL [CL03], BBS [ASM06, CDL16, TZ23], or struc-ture-preserving signatures (SPS) [AFG+10], which admit efficient proof systems.
- At the same time, we aim to achieve post-quantum *privacy*, even if the underlying signature can be forged by a quantum computer. Indeed, if privacy of the tokens does not hold against quantum attacks, malicious actors may collect already spent tokens ahead of time *today* and use quantum computers (once available) to link transactions afterwards.

Everlasting Anonymous Rate-Limited Tokens. To implement the above goals, we will not rely on any post-quantum assumption for privacy. Instead, we ask whether we can achieve the even stronger notion of privacy against *unbounded adversaries*. Concretely, we construct what we refer to as *Everlasting Anonymous Rate-Limited Tokens* (EARLT), achieving similar functionality to CHKLM e-tokens, but with everlasting anonymity guarantee and with some changes in the security definitions.

More precisely, in an initial interaction with the issuer, the user obtains what we call a *token dispenser*, parametrized by two parameters N and k, where $N \leqslant k$. The token dispenser will allow the user to generate

Approach	Runtime†				Size (# group elem. & scalars)	
	User	I	Show	V	|D|	$\|\tau\|$
CHKLM [CHK+06]	$\mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$	$\log N \cdot \mathsf{poly}(\lambda)$	$\log N \cdot \mathsf{poly}(\lambda)$	$O(1)$	$O(\log\log N)$
Naive: deg k polynomial	$k \cdot \mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$	$(k + \log N) \cdot \mathsf{poly}(\lambda)$	$\log N \cdot \mathsf{poly}(\lambda)$	$O(k)$	$O(\log\log N)$
EARLT (Sec. 5)	$k \log^2 k \cdot \mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$	$\log N \cdot \mathsf{poly}(\lambda)$	$\log N \cdot \mathsf{poly}(\lambda)$	$O(k)$	$O(\log \lambda)$

Fig. 1. Comparison on efficiency metrics of the described approaches. Note that anonymity of the CHKLM construction is *computational*, while others are *everlasting*. The runtime columns denote the running time for the User (User), Issuer (I), Showing (Show), and Verification (V) algorithms. The sizes are measured in the number of group elements and scalars. Denote k as the total token limit per dispenser before anonymity is not guaranteed (does not apply for the CHKLM approach), and N as the limit on tokens per context string. †: The polynomials in poly are different for each quantity.

(or "show") publicly verifiable tokens, each of them associated with a context string ctxt. Then, as long as a particular token dispenser has not been used to show more than k tokens, and no more than N tokens have been shown for a particular context, no unbounded adversary will be able to link token showings with each other or link them to a particular issuance session. Further, any set consisting of more than N tokens generated by the same dispenser and associated to the same context can be used to reconstruct the identity of the user.

Our construction is based on bilinear pairing groups, mainly utilizing KZG commitments [KZG10], SPS schemes [AFG+10], and Groth-Sahai proofs [GS08]. Crucially, our issuance protocol only requires communication and computational costs for the issuer that is independent of k, making it more ideal than issuing multiple one-time-use tokens, e.g., as in Privacy Pass.

We give technical details in the overview below. However, we stress here that our construction is non-trivial. One could indeed attempt to instantiate the CHKLM e-token construction with a k-wise independent function (more specifically, a random degree k polynomial) instead of the PRF, but as we explain below, this would lead to a construction where showing a token requires time that grows (linearly) with k. Instead, we are going to propose a construction where the showing time is *independent* of k. In Figure 1, we give an overview of the asymptotic parameters of our construction in comparison with the CHKLM construction (without everlasting privacy) and the naive construction using polynomials. We note that our dependence on k at issuance is somewhat inherent.

We view our work as a first step towards *concretely practical* rate-limited anonymous tokens with everlasting anonymity. Rate-limited tokens have an ongoing standardization effort [YW25] whose candidate is only privately-verifiable and computationally anonymous. Our construction is publicly verifiable and everlasting anonymous, while being *asymptotically* efficient. We also give concrete efficiency comparison with lattice-based alternatives in Section 5.3.

1.1 Technical Overview

Before going into the technical details of the construction, we briefly discuss the syntax and security of rate-limited tokens.

Everlasting anonymous rate-limited tokens. An EARLT scheme cons-ists of the issuer with their issuer's keys $(\mathsf{sk}_\mathsf{I}, \mathsf{pk}_\mathsf{I})$, the user(s) with their user keys $(\mathsf{sk}_\mathsf{User}, \mathsf{pk}_\mathsf{User})$, and the verifier(s). The users are identified through their public keys. The primitive consists of the following components:

- **An issuance protocol** allowing the user to obtain a token dispenser D certified by the issuer's secret key.
- **Showing and verification:** The user, given their own dispenser D, a nonce r and a context ctxt, generates a token τ with serial number sn that can be verified using the issuer's public key.

- **Identifying/Linking** a double-spender from two tokens for the same context with identical serial number.

We now discuss the required security properties and refer to Section 3 for more details and distinction from prior definitions.

- **Everlasting anonymity:** No *unbounded* adversarial collusion of issuers and verifiers can link tokens with each other or with an issuance session, if no dispenser is used more than the per-context limit N or the total limit k.
- **Unforgeability:** ensures that no adversary who is issued Q dispensers can produce more than $Q \cdot N$ tokens with distinct serial numbers that are valid for a particular context ctxt.
- **Linkability:** ensures that the verifier can detect and identify a double-spender. In particular, no adversary who can request *multiple token dispensers* can find two tokens such that (1) they verify with the same context and different nonces, (2) their serial numbers are identical, and (3) the identification algorithm returns a public key that does not correspond to any issuance session. Our linkability notion strengthens the definition in [CHK+06], which only considers adversaries with *one issued dispenser*.
- **Exculpability:** No malicious users, *even when colluding with the issuer*, can frame honest users as a double-spender. Framing, in this context, means forcing the identification algorithm to output an honest user's public key.

CHKLM RATE-LIMITED TOKENS. As mentioned earlier, the CHKLM [CHK+06] approach relies on a signature scheme, a PRF, and zero-knowledge proofs. In particular, the dispenser contains the secret key $\mathsf{sk}_{\mathsf{User}}$, a PRF seed s, and a signature σ on $(\mathsf{sk}_{\mathsf{User}}, s)$. To show a token for a context string ctxt and nonce $r \in \mathbb{Z}_p$, the user increments a counter cnt (i.e., number of tokens already shown for ctxt), and computes the serial number and double-spending equation (which will be used to identify a double-spending user) as follows

$$\mathsf{sn} = F(s, (0||\mathsf{ctxt}||\mathsf{cnt})), \mathsf{dbsp} = r \cdot F(s, (1||\mathsf{ctxt}||\mathsf{cnt})) + \mathsf{pk}_{\mathsf{User}} .$$

The token consists of $\mathsf{sn}, \mathsf{dbsp}$ and a proof π proving knowledge of $\mathsf{sk}_{\mathsf{User}}, s, \sigma$ and cnt such that (1) σ is valid for $(\mathsf{sk}_{\mathsf{User}}, s)$ with respect to pk_{I}, and (2) sn and dbsp are computed as above, and (3) $\mathsf{cnt} \in [0, N-1]$. For identification, assuming that one has two tokens computed from the same $\mathsf{sk}_{\mathsf{User}}, s, \mathsf{ctxt}, \mathsf{cnt}$ but with different nonces $r \neq r'$, the serial number would be identical and solving simple linear equation on dbsp will reveal $\mathsf{pk}_{\mathsf{User}}$. Since one can use statistical zero-knowledge proofs, the major roadblock to achieving everlasting anonymity is the reliance on the PRF. In particular, sn and dbsp look pseudorandom only against computationally bounded adversaries.

NAIVE CONSTRUCTION: DEGREE-k POLYNOMIALS. Our first attempt in such a construction is to replace the PRF with a k-wise independent function family, namely a random $\mathbb{Z}_p$-polynomial of degree k. In particular, the user samples a random polynomial f of degree k and commits to f via a KZG commitment [KZG10] C_f. This commitment is then signed along with $\mathsf{pk}_{\mathsf{User}}$ by the issuer via a structure-preserving signature.

To show a token for a context string ctxt, the user increments a counter $\mathsf{cnt} \in [0, N-1]$, keeps track of the number of times a token has been shown for this context, and computes the serial number $\mathsf{sn} = f(x)$ where $x = \mathsf{ctxt} \cdot N + \mathsf{cnt}$ (assuming $0 \leqslant x < p$). Then, it computes an opening of the KZG commitment with respect to the evaluation of x as $f(x)$ and provides a zero-knowledge proof attesting that it knows a counter, a KZG commitment of f, a valid signature of the commitment, and an opening of the commitment with respect to x and $\mathsf{sn} = f(x)$. As the verification of KZG and SPS can be written as pairing product equations, the proof can be efficiently instantiated with Groth-Sahai proofs [GS08].

Double-spending detection can be incorporated by having the user sample an additional polynomial f', similarly commit it as $C_{f'}$, and have it signed along with $(\mathsf{pk}_{\mathsf{User}}, C_f)$. The value dbsp is computed as $r \cdot f'(x) + \mathsf{sk}_{\mathsf{User}}$ and tracing can be done similarly to CHKLM approach.

However, the showing of this construction runs in time linear in k. This is due to (1) evaluating the degree k polynomials f, f' and (2) computing the openings of the KZG commitments. Since the user does not know in advance which context it will show a token for, neither (1) nor (2) can be precomputed. In general, this makes the showing (which in applications is performed very frequently) as costly as the user's computation

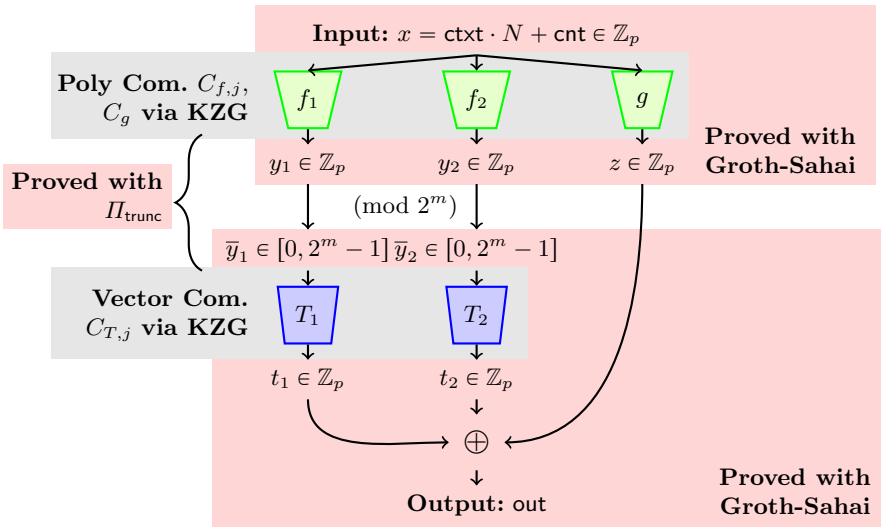

Fig. 2. Our particular instantiation of Pagh-Pagh function family. The green and blue boxes denote the keys for the function with f_1, f_2, g being polynomials of degree $d = \Theta(\lambda)$ and T_1, T_2 be tables of size $S = \Theta(k)$. The gray boxes denotes how each part is committed to by the user at issuance time. The pink boxes denote how to prove the correctness of each computation step.

---

at issuance (which is only done once). This is prohibitively inefficient for scenarios where the user needs to produce a large number of tokens within a short time frame (e.g., accessing multiple subscription services at once, or making multiple payments at a time). See Figure 1 for comparison with our solution below.

Our solution: (almost) k-independent functions with locality. We ideally want the showing runtime to be independent of the parameter k. To achieve this more stringent efficiency requirement, we instantiate the k-wise independent function with the Pagh-Pagh [PP08, BHKN19] function family instead of degree k polynomials. Each function $F_{\mathsf{key}} : \mathcal{D} \to \mathcal{R}$ (where $\mathcal{R}$ is a group) is defined by a key key consisting of d-wise independent functions $h_1, h_2 : \mathcal{D} \to [0, S-1], g : \mathcal{D} \to \mathcal{R}$ and uniformly random tables $T_1, T_2 \in \mathcal{R}^S$ (i.e. uniformly random $\mathcal{R}^S$ vectors). The function F_{key} is computed via $F_{\mathsf{key}}(x) = T_1[h_1(x)] + T_2[h_2(x)] + g(x)$. For $d = \Theta(\lambda)$ and $S = \Theta(k)$, no *unbounded* adversary, making k adaptive queries to an evaluation oracle, can distinguish it from a random function with non-negligible advantage (in λ).

For compatibility with prime-order group structure, we particularly instantiate this function family by: (a) Set $\mathcal{D} = \mathcal{R} = \mathbb{Z}_p$ and $S = 2^m = \Theta(k)$ for some integer m, (b) Define the functions $h_1, h_2 : \mathbb{Z}_p \to [0, S-1]$ with uniformly random polynomials $f_1, f_2 \in \mathbb{Z}_p^{\leq d}[\mathsf{X}]$ where $h_j(x) = (f_j(x) \pmod S)$, and (c) Set g to be a uniformly random degree d in $\mathbb{Z}_p^{\leq d}[\mathsf{X}]$.[4] Evaluating the function takes time *linear in* $d = \Theta(\lambda)$*, independent of* k, and the key size is *only a constant factor larger than the naive approach.*

To efficiently instantiate our solution with this function, we modify the following components from the naive construction. (Also, see the figure summarizing our approach in Figure 2, and refer to Sections 4 and 5 for more details in our building blocks and construction, respectively.):

- **Commitments to the function key.** We individually commit to the polynomials f_1, f_2, g as $C_{f,1}, C_{f,2}$, C_g via KZG. To commit to tables T_1, T_2 as $C_{T,1}, C_{T,2}$, we also use KZG but as a vector commitment scheme, i.e., by committing to a polynomial interpolating the value at each position $i \in [0, S-1]$. Here, $C_{T,j}$ are only opened on positions $i \in [0, S-1]$, so the user can precompute all the openings at issuance time to save cost during showing—this can be done in $O(S \log^2 S)$ group exponentiations for certain (but large and widely available) class of groups [FK23].

[4] The readers may notice that f_1, f_2 are not k-wise independence, but almost k-wise independence. This however only negligibly affects the security of the function family. We refer to Section A for further discussion.

- **Proof of correct evaluation.** To prove that sn and dbsp are computed correctly with respect to the signed commitments, we consider the intermediate values $y_1, y_2, \overline{y}_1, y_2, z, t_1, t_2$ as illustrated in Figure 2. Then, we commit to each of the intermediate values and the KZG commitments using Groth-Sahai commitments and prove the following constraints with respect to the committed values:
  - **Polynomials f_1, f_2, g are evaluated correctly.** As in the naive construction, the user computes the corresponding KZG openings to $C_{f,j}$ and C_g and relies on Groth-Sahai to prove knowledge of such openings.
  - **Correct positions of T_1, T_2 are used.** The user takes the precomputed openings of $C_{T,j}$ and proves knowledge of it analogously.
  - **Outputs of f_1, f_2 are truncated correctly.** Here, we need to show that the evaluation $y_j = f_j(x)$ and the index $\overline{y}_j \in [0, S-1]$ are such that $\overline{y}_j = y_j \pmod S$. Since we set $S = 2^m$ for some $m \in \mathbb{N}$, this is equivalent to proving that the scalar y_j when written as an integer in $[0, p-1]$ and truncated to an m-bit integer is equal to $\overline{y}_j$. Here, we rely on a special purpose non-interactive ZK proof system Π_{trunc}, which we detail in Section 5.2, to prove that the truncation is done correctly.
    As an intuition, the user proves that there exists a bit-decomposition of y_j denoted $(b_0, \ldots, b_n)$ where $n = \lfloor \log p \rfloor$ such that $y_j = \sum_{i=0}^{n} b_i 2^n$ and $\overline{y}_j = \sum_{i=0}^{m-1} b_i 2^i$. However, this is insufficient, because in the field $\mathbb{Z}_p$ (the underlying field our proof system is working with), the n-bit decomposition of y_j can be ambiguous, i.e., there can be two possible n-bit decompositions of y_j due to the wrap-around modulo p. We overcome this issue by identifying the necessary and sufficient conditions for this particular statement and give a proof system by using similar techniques from Bulletproof [BBB+18] and Compressed Σ-protocols [AC20]. Efficiency-wise this proof is comparable to range proofs, which are already used in the CHKLM protocol to show that $\mathsf{cnt} \in [0, N-1]$.

Security. Finally, we provide some intuition on why the sketched scheme (with some modifications) satisfies each required security property. For the concrete security statements, refer to Section 6.

Everlasting anonymity. In addition to pseudorandomness of the selected function family against any unbounded adversary, we also need that (1) the commitments to each component of the function key are statistically hiding and (2) the proof systems we use are statistically zero-knowledge. For (1), we rely on the perfectly hiding variant of KZG [KZG10]. For (2), we can generate the CRS of Groth-Sahai proofs in the perfectly hiding mode. Concerning the proof system Π_{trunc}, we provided an instantiation where the zero-knowledge guarantee does not rely on the random oracle, but is in fact statistical. This can be achieved via techniques from e.g. [Gro04, Dam00, CDS94] (also see Section 5.2).

Unforgeability. At a high-level, if an adversary outputs more valid tokens with distinct serial numbers for a context string ctxt than allowed (i.e., number of issuance queries Q_{Iss} times number of tokens allowed per context $N = 2^{\ell_{\mathsf{cnt}}}$), then by proof of knowledge property, we can extract the underlying witnesses containing (a) the tuple of commitments and public key, (b) a valid signature on this tuple, and (c) intermediate values and corresponding openings to each evaluation step (including the counter). Now, one of the the following must occur:

(1) One of the extracted tuple of commitments was not signed at issuance time. In this case, we break unforgeability of the SPS scheme.

(2) All $Q_{\text{Iss}} \cdot N + 1$ extracted tuple of commitments are signed. Note that the counter $\mathsf{cnt} \in [0, N-1]$ is also extracted. Therefore, by the Pigeonhole principle, there are at least two tokens with the same extracted tuple of commitments and counter. Since the serial numbers of these tokens are different, the adversary either breaks soundness of the proof systems or binding of the KZG commitments. Otherwise, the serial numbers should be identical.

Here, the intermediate values and the openings of KZG commitments contain $\mathbb{Z}_p$-scalars, but the extractability of Groth-Sahai proofs only allows extracting group elements (if a witness a is in $\mathbb{Z}_p$, we extract aG_1 instead). Hence, we additionally require the user to prove knowledge of the underlying discrete logarithms. This raises additional challenges in the proof provided in Section 6.2.

Linkability. If an adversary can double-spend, i.e., producing *two tokens* with the same serial number $\mathsf{sn}_0 = \mathsf{sn}_1$ that are both valid for the same context ctxt but for different nonces $r_0 \neq r_1$, we want the identification algorithm to output one of the adversarial users' public keys. Assume towards contradiction

that none of the adversarial users can be linked. Proceeding as in the unforgeability proof, we extract the underlying witnesses and also rule out the case where the extracted tuple of commitments and public key is not signed. Hence, we only need to consider the following three cases:

(1) For both tokens, the extracted commitments and public key are from the *same issuance session* and the extracted counters $\mathsf{cnt}_0 = \mathsf{cnt}_1$ are *identical*
(2) For both tokens, the extracted commitments and public key are from the *same issuance session* but the extracted counters $\mathsf{cnt}_0 \neq \mathsf{cnt}_1$ are *distinct*.
(3) The extracted commitments and public key are from *different issuance sessions*.

We rule out case (1) using binding of KZG and soundness of the proof systems as in the argument for unforgeability. Case (2) corresponds to the adversary finding a collision in the committed function, i.e., we have two inputs $(\mathsf{ctxt}||\mathsf{cnt}_0), (\mathsf{ctxt}||\mathsf{cnt}_1)$ evaluating to the same $\mathsf{sn}_0 = \mathsf{sn}_1$. In the CHKLM construction, this case does not occur as they use DY-PRF [DY05], which is a bijection from $\mathbb{Z}_p$ to $\mathbb{G}$. To ensure that this event does not occur (barring some negligible probability), we make the following modifications: (a) we enlarge the range of the function so that the serial number lies in $\mathbb{Z}_p^2$ and (b) instead of computing $\mathsf{sn} = F_{\mathsf{key}}(\mathsf{ctxt}||\mathsf{cnt})$, we compute $\mathsf{sn} = F^*_{\mathsf{key},\boldsymbol{\gamma}_0,\boldsymbol{\gamma}_1}(\mathsf{ctxt}||\mathsf{cnt}) = F_{\mathsf{key}}(\mathsf{ctxt}||\mathsf{cnt}) + \mathsf{cnt}\cdot\boldsymbol{\gamma}_0 + \boldsymbol{\gamma}_1$, where $\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1 \in \mathbb{Z}_p^2$ are randomization factors sampled by the issuer after the user sends the commitments to the keys. Then, except for negligible probability over $\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1$, via simple union bound, this function has no collisions. This modification also allows us to deal with case (3) where the adversary finds two inputs that make the outputs of two functions collide.

The actual proof, given in Section 6.3, is more complex than this outline. In particular, the sketched argument for (2) and (3) works if we make an assumption that key is fixed at issuance when the adversary sends the commitments. This, however, is not necessarily the case as the KZG commitments are only computationally binding and the events above do not immediately imply breaking evaluation-binding of KZG. Still, it is possible to reduce to a stronger binding property for KZG (which we defined in Section 2), implied by the recently proposed falsifiable ARSDH assumption [LPS24](and we show such implication in Section B.1).

Exculpability. Exculpability is implied by everlasting anonymity and the discrete logarithm assumption. In a nutshell, everlasting anonymity ensures that the tokens reveal no information about the honest users; hence, the only way to frame an honest user is by extracting the secret key of this user, breaking the DLOG assumption.

2 Preliminaries

NOTATIONS. We denote $[n, m] = \{n, n+1, \ldots, m\}$ for any integers n, m where $n \leqslant m$ and $[n] = [1, n]$ for any positive integer n. Throughout the paper, we use λ as the security parameter. Denote $x \leftarrow a$ as assigning value a to a variable x. Denote $a \leftarrow\!\!\$\ S$ as uniformly sampling a from a finite set S. We 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 of A; $(y_1, y_2) \leftarrow\!\!\$\ \langle \mathsf{A}(x_1) \rightleftharpoons \mathsf{B}(x_2)\rangle$ denotes a pair of interactive algorithms A, B with inputs x_1, x_2 and outputs y_1, y_2, respectively. We denote formal variables in polynomials with sans-serif letters (e.g., X, Y). For any prime modulus p, let $\mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$ denote the ring of $\mathbb{Z}_p$-polynomials with degree at most d. We might refer to polynomial $g(\mathsf{X})$ using the shorthand g when it is clear from the context. We often denote vectors using bold-sized letters (e.g., $\boldsymbol{v}, \boldsymbol{H}$). We denote $\mathsf{Adv}^{\mathsf{game}}_{\Pi}(\mathcal{A}, \lambda)$ as the advantage of an adversary $\mathcal{A}$ in the security game game for a primitive/scheme Π.

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 all generators of $\mathbb{G}$. We write group elements and scalar with upper-case and lower-case letters, resp. 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 input 1^λ and outputting $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$ such that $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of λ-bit prime order p, and $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. In particular, the map e satisfies (1) *bilinearity*, i.e., for any $A \in \mathbb{G}_1, B \in \mathbb{G}_2$ 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 $G_1 \in \mathbb{G}_1^*, G_2 \in \mathbb{G}_2^*$, $\mathsf{e}(G_1, G_2) \in \mathbb{G}_T^*$. We assume that the group descriptions contain the generators $G_1 \in \mathbb{G}_1^*, G_2 \in \mathbb{G}_2^*, G_T \in \mathbb{G}_T^*$.

CRYPTOGRAPHIC ASSUMPTIONS. In this work, we consider the DLOG assumption, SXDH assumption (which is DDH assumed over the two source groups), q-SDH assumption, and q-ARSDH assumption (recently proposed in [LPS24]) with the games defined in Figure 3. Note that q-ARSDH assumption implies q-SDH assumption, which implies the DLOG assumption. For DLOG, q-SDH, and q-ARSDH games, we denote the corresponding advantage as

$$\mathsf{Adv}^{\text{dlog}/q\text{-sdh}/q\text{-arsdh}}_{\mathsf{GGen}}(\mathcal{A},\lambda) := \mathsf{Pr}[(\text{DL}/q\text{-SDH}/q\text{-ARSDH})^{\mathcal{A}}_{\mathsf{GGen}}(1^\lambda) = 1] \ .$$

For the DDH game on $\mathbb{G}_t$ for $t \in \{1,2\}$ and the SXDH assumption, we have that

$$\mathsf{Adv}^{\text{ddh}}_{\mathsf{GGen},t}(\mathcal{A},\lambda) := |\mathsf{Pr}[\text{DDH}^{\mathcal{A}}_{\mathsf{GGen},0,t}(1^\lambda) = 1] - \mathsf{Pr}[\text{DDH}^{\mathcal{A}}_{\mathsf{GGen},1,t}(1^\lambda) = 1]| \ ,$$
$$\mathsf{Adv}^{\text{sxdh}}_{\mathsf{GGen}}(\mathcal{A},\lambda) := \mathsf{Adv}^{\text{ddh}}_{\mathsf{GGen},1}(\mathcal{A},\lambda) + \mathsf{Adv}^{\text{ddh}}_{\mathsf{GGen},2}(\mathcal{A},\lambda) \ .$$

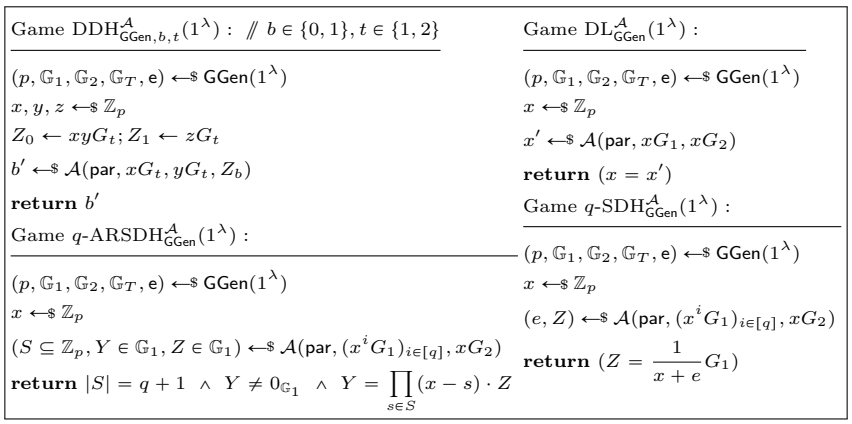

Fig. 3. Games for each group-based assumptions.

INTERACTIVE PROOFS. An interactive proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{P}, \Pi.\mathsf{V})$ for a family of relations $\mathcal{R}$ with the following syntax:

- $\mathsf{crs} \leftarrow\!\$\ \Pi.\mathsf{Setup}(1^\lambda)$ generates the crs which also defines the relation $\mathcal{R} = \mathcal{R}_{\mathsf{crs}}$. We will often omit the subscript crs when clear from the context. We also denote the induced language of $\mathcal{R}$ as $\mathcal{L}_{\mathcal{R}} := \{\mathbb{x} : \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$.
- $(\perp, 0/1) \leftarrow\!\$\ \langle \Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \rightleftharpoons \Pi.\mathsf{V}(\mathsf{crs}, \mathbb{x})\rangle$ is a $(2\mu+1)$-move protocols between the prover and the verifier.

An interactive proof system is public-coin if the randomness of the verifier is made public. We recall the definition for tree of transcripts from [AFK22].

Definition 2.1 (Tree of transcripts [AFK22]). *Let $k_1, \ldots, k_\mu \in \mathbb{N}$. A $(k_1, \ldots, k_\mu)$-tree of transcripts for a $(2\mu+1)$-move public-coin interactive proof Π is a set of $K = \prod_{i=1}^{\mu} k_i$ transcripts arranged in the following tree structure: The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf node.*

We require that the proof system satisfies the following properties.

Correctness. For any $\mathsf{crs} \in [\Pi.\mathsf{Setup}(1^\lambda)]$ and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $\langle \Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \rightleftharpoons \Pi.\mathsf{V}(\mathsf{crs}, \mathbb{x})\rangle$ always output 1.

$(k_1, \ldots, k_\mu)$**-out-of-**$(N_1, \ldots, N_\mu)$ **Special-soundness.** For a relaxed relation $\widetilde{\mathcal{R}} \supseteq \mathcal{R}$, given a $(k_1, \ldots, k_\mu)$-tree of valid transcripts for Π, there exists an extractor Ext which extracts a valid witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \widetilde{\mathcal{R}}$.

Special honest-verifier zero-knowledge. There exists a simulator Sim such that for any $\mathsf{crs} \in [\Pi.\mathsf{Setup}(1^\lambda)]$, $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and randomness ρ for V, the distributions of transcripts trans from $\langle \Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \rightleftharpoons \Pi.\mathsf{V}(\mathsf{crs}, \mathbb{x}; \rho)\rangle$ and $\mathsf{Sim}(\mathsf{crs}, \mathbb{x}, \rho)$ are identical.

Zero-knowledge. There exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{\mathsf{P}})$ such that (a) the CRS generated from $(\mathsf{crs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda)$ is indistinguishable from $\Pi.\mathsf{Setup}$, i.e., the following advantage is bounded

$$\mathsf{Adv}^{\mathsf{dist}}_{\Pi,\mathsf{Sim}}(\mathcal{A}, \lambda) := |\mathsf{Pr}[\mathcal{A}(\mathsf{crs}) = 1 | \mathsf{crs} \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda)] - \mathsf{Pr}[\mathcal{A}(\mathsf{crs}) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda)]| \ ,$$

and (b) no malicious verifier $\mathcal{A}$ can distinguish between interacting with a prover $\Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$ and a simulator $\mathsf{Sim}(\mathsf{crs}, \mathsf{td}, \mathbb{x})$, in particular, the following advantage is bounded

$$\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathsf{Sim}}(\mathcal{A}, \lambda) := \left| \mathsf{Pr}\left[\begin{array}{l|l} (\mathbb{x}, \mathbb{w}) \in \mathcal{R} & (\mathsf{crs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda) \\ \wedge\ b = 1 & (\mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}) \\ & (\bot, b) \leftarrow\!\!\$\ \langle \Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \rightleftharpoons \mathcal{A}(\mathsf{st})\rangle \end{array} \right] - \right.$$
$$\left. \mathsf{Pr}\left[\begin{array}{l|l} (\mathbb{x}, \mathbb{w}) \in \mathcal{R} & (\mathsf{crs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda) \\ \wedge\ b = 1 & (\mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}) \\ & (\bot, b) \leftarrow\!\!\$\ \langle \mathsf{Sim}_{\mathsf{P}}(\mathsf{crs}, \mathsf{td}, \mathbb{x}) \rightleftharpoons \mathcal{A}(\mathsf{st})\rangle \end{array} \right] \right| \ ,$$

NON-INTERACTIVE ZERO-KNOWLEDGE PROOFS. A non-interactive zero knowledge (NIZK) proof system Π for a family of languages $\mathcal{R}$ consists of the followingalgorithms:

- $\mathsf{crs} \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda)$ generates the CRS which implicitly defines the relation $\mathcal{R} = \mathcal{R}_{\mathsf{crs}}$ (again omitting the subscript when clear from the context) and the induced language $\mathcal{L}_{\mathcal{R}}$.
- $\pi \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$ on input $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ outputs a proof π.
- $0/1 \leftarrow \Pi.\mathsf{V}(\mathsf{crs}, \mathbb{x}, \pi)$ checks whether the proof is valid.

We require that the proof system satisfies the following properties:

Correctness: For any $\mathsf{crs} \in [\Pi.\mathsf{Setup}(1^\lambda)]$, $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the following procedure always return 1.

$$\pi \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \textbf{return}\ \Pi.\mathsf{V}(\mathsf{crs}, \mathbb{x}, \pi) \ .$$

Soundness: For any adversary $\mathcal{A}$, the soundness advantage of $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{\mathsf{sound}}_{\Pi}(\mathcal{A}, \lambda) := \mathsf{Pr}\left[\mathbb{x} \notin \mathcal{L}_{\mathcal{R}} \ \wedge\ \Pi.\mathsf{V}(\mathsf{crs}, \mathbb{x}, \pi) = 1 \middle| \begin{array}{l} \mathsf{crs} \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda) \\ (\mathbb{x}, \pi) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}) \end{array} \right] \ .$$

Zero-knowledge: There exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{\mathsf{P}})$ such that the following ZK advantage of any adversary $\mathcal{A}$ is bounded, i.e.,

$$\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathsf{Sim}}(\mathcal{A}, \lambda) := |\mathsf{Pr}[\mathcal{A}^{\mathcal{O}_0}(\mathsf{crs}) = 1 | \mathsf{crs} \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda)] - \mathsf{Pr}[\mathcal{A}^{\mathcal{O}_1}(\mathsf{crs}) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda)]| \ ,$$

where the oracle $\mathcal{O}_b(\mathbb{x}, \mathbb{w})$ first checks if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and abort if not; then if $b = 0$, it returns $\pi \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$; otherwise, it returns $\pi \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{P}}(\mathsf{td}, \mathbb{x})$. Note that throughout the paper, we will only consider NIZKs with statistical (or perfect) zero-knowledge, i.e., ones where $\mathcal{A}$ is unbounded; moreover, we do not assume hash functions are random oracles in our ZK guarantees.

Adaptive knowledge-soundness: We adapt the definition of adaptive knowledge-soundness from [AFK22] which is defined specifically for proof systems relying on random oracles. In particular, a non-interactive random oracle proof Π is adaptively knowledge sound for a relaxed relation $\widetilde{\mathcal{R}} \supseteq \mathcal{R}$ with knowledge error $\kappa : \mathbb{N} \times \mathbb{N} \to [0, 1]$, if there exists a positive polynomial q and a knowledge extractor Ext such that for any adversary $\mathcal{A}$ (taking input crs, making at most $Q_{\mathsf{H}} = Q_{\mathsf{H}}(\lambda)$ queries to H, and returns $(\mathbb{x}, \pi)$), then $\mathsf{Ext}^{\mathcal{A}}(\mathsf{crs})$ runs in expected number of steps that is polynomial in λ and Q_{H} and outputs a tuple $(\mathbb{x}, \pi, \mathsf{aux}, v; \mathbb{w})$ such that

- The following distributions over the randomness of $\mathcal{A}$, H, Ext and the setup algorithm

$$\left\{(\mathbb{x},\pi,\mathsf{aux},v):\begin{array}{l}\mathsf{crs}\leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda);\\ (\mathbb{x},\pi,\mathsf{aux})\leftarrow\!\!\$\ \mathcal{A}^{\mathsf{H}}(\mathsf{crs});\\ v\leftarrow\Pi.\mathsf{V}^{\mathsf{H}}(\mathsf{crs},\mathbb{x},\pi)\end{array}\right\}\ \text{ and }\ \left\{(\mathbb{x},\pi,\mathsf{aux},v):\begin{array}{l}\mathsf{crs}\leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda);\\ (\mathbb{x},\pi,\mathsf{aux},v;\mathbb{w})\leftarrow\!\!\$\ \mathsf{Ext}^{\mathcal{A}}(\mathsf{crs})\end{array}\right\},$$

  are identical.
- Let $\varepsilon(\lambda):=\mathsf{Pr}[\Pi.\mathsf{V}^{\mathsf{H}}(\mathsf{crs},\mathbb{x},\pi)=1:\mathsf{crs}\leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda);(\mathbb{x},\pi,\mathsf{aux})\leftarrow\!\!\$\ \mathcal{A}^{\mathsf{H}}(\mathsf{crs})]$. Then,

$$\mathsf{Pr}\left[v=1\ \wedge\ (\mathbb{x},\mathbb{w})\in\widetilde{\mathcal{R}}\middle|\begin{array}{l}\mathsf{crs}\leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda)\\ (\mathbb{x},\pi,\mathsf{aux},v;\mathbb{w})\leftarrow\!\!\$\ \mathsf{Ext}^{\mathcal{A}}(\mathsf{crs})\end{array}\right]\geqslant\frac{\varepsilon(\lambda)-\kappa(\lambda,Q_{\mathsf{H}})}{q(\lambda)}\ .$$

  Here, Ext implements the RO H for $\mathcal{A}$ and the randomness is over the random coins of Ext and $\mathcal{A}$.

Polynomial commitments. A polynomial commitment scheme PCS for polynomials in $\mathbb{Z}_p[\mathsf{X}]$ of bounded degree d consists of algorithms

- $\mathsf{crs}_{\mathsf{PCS}}\leftarrow\!\!\$\ \mathsf{PCS.Setup}(1^\lambda,d)$ a setup algorithm,where the CRS also defines the modulus p.
- $(C,\rho)\leftarrow\!\!\$\ \mathsf{PCS.Com}(\mathsf{crs}_{\mathsf{PCS}},f\in\mathbb{Z}_p^{\leqslant d}[\mathsf{X}])$ computes a commitment C of f and a state ρ.
- $\mathsf{open}\leftarrow\!\!\$\ \mathsf{PCS.Open}(\mathsf{crs}_{\mathsf{PCS}},f,\alpha,\beta,C,\rho)$ computes an opening that C commits to f such that $f(\alpha)=\beta$.
- $0/1\leftarrow\mathsf{PCS.V}(\mathsf{crs}_{\mathsf{PCS}},C,\alpha,\beta,\mathsf{open})$ verifies that the opening is valid.

We consider the following properties of PCS.

Correctness. For any $d=d(\lambda)$, $\mathsf{crs}\in[\mathsf{PCS.Setup}(1^\lambda,d)]$, $f\in\mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$ and $\alpha\in\mathbb{Z}_p$, the following procedure always returns 1.

$$\begin{array}{l}(C,\rho)\leftarrow\!\!\$\ \mathsf{PCS.Com}(\mathsf{crs}_{\mathsf{KZG}},f);\mathsf{open}\leftarrow\!\!\$\ \mathsf{PCS.Open}(\mathsf{crs},C,f,\alpha,f(\alpha),\rho)\\ \textbf{return}\ \mathsf{PCS.V}(\mathsf{crs},C,\alpha,f(\alpha),\mathsf{open})=1\end{array}$$

Evaluation-binding. For $d=d(\lambda)$, the evaluation-binding advantage of any adversary $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{PCS},d}(\mathcal{A},\lambda):=\mathsf{Pr}\left[\begin{array}{l}\beta_1\neq\beta_2\ \wedge\\ \forall i\in[2]:\mathsf{PCS.V}(\mathsf{crs},C,\alpha,\beta_i,\mathsf{open}_i)=1\end{array}\middle|\begin{array}{l}\mathsf{crs}\leftarrow\!\!\$\ \mathsf{PCS.Setup}(1^\lambda,d)\\ (C,\alpha,(\beta_i,\mathsf{open}_i)_{i\in[2]})\leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs})\end{array}\right].$$

Perfectly hiding. For any $d=d(\lambda)$, $\mathsf{crs}\in[\mathsf{PCS.Setup}(1^\lambda,d)]$, and $f_0,f_1\in\mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$, the following distributions $\mathcal{D}_0,\mathcal{D}_1$ are identical

$$\mathcal{D}_b:=\{C:(C,\rho)\leftarrow\!\!\$\ \mathsf{PCS.Com}(\mathsf{crs},f_b)\}\ .$$

Degree-binding. We additionally define the degree-binding property for polynomial commitment. The property says that no adversary can compute a commitment and $d+2$ openings such that all the openings verify but the opened evaluation points do not lie in a degree $\leqslant d$ polynomial. This is similar to the strong correctness property defined in [KZG10], with the distinction that we allow the adversary in our case to pick the evaluation points instead of the game randomly sampling them. The property is also similar to function-binding defined for functional commitments in [LM19].

For $d=d(\lambda)$, the degree-binding advantage of any adversary $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{PCS},d}(\mathcal{A},\lambda):=\mathsf{Pr}\left[\begin{array}{l}(\forall i\in[d+2]:\\ \mathsf{PCS.V}(\mathsf{crs},C,\alpha_i,\beta_i,\mathsf{open}_i)=1)\ \wedge\\ (\forall f\in\mathbb{Z}_p^{\leqslant d}[\mathsf{X}],\exists i\in[d+2]:f(\alpha_i)\neq\beta_i)\end{array}\middle|\begin{array}{l}\mathsf{crs}\leftarrow\!\!\$\ \mathsf{PCS.Setup}(1^\lambda,d)\\ (C,(\alpha_i,\beta_i,\mathsf{open}_i)_{i\in[d+2]})\leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs})\end{array}\right].$$

VECTOR COMMITMENTS. A vector commitment scheme VC for a vector with elements in $\mathcal{M}$ consists of the following algorithms:

- $\mathsf{crs}_{\mathsf{VC}} \leftarrow\!\!\$\ \mathsf{VC.Setup}(1^\lambda, S)$ generates the CRS which defines $\mathcal{M} = \mathcal{M}_{\mathsf{crs}}$.
- $(C, (\mathsf{open}_i)_{i\in[0,S-1]}) \leftarrow\!\!\$\ \mathsf{VC.Com}(\mathsf{crs}_{\mathsf{VC}}, \boldsymbol{m} \in \mathcal{M}^S)$ commits to the vector $\boldsymbol{m}$ as C and compute the openings to each position $i \in [0, S-1]$.
- $0/1 \leftarrow \mathsf{VC.V}(\mathsf{crs}_{\mathsf{VC}}, C, i, m_i, \mathsf{open}_i)$ verifies that C commits to a $\boldsymbol{m}$ such that the i-th position is m_i.

We do not define an explicit opening algorithm, and assume that the openings for all positions are computed along with the commitment. Note that for the context of vector commitment schemes, we index elements starting from 0 – for convenience in defining proof systems later on. However, throughout the paper, we will index vectors starting from 1. We consider the following properties of vector commitments

Correctness. For any $\mathsf{crs} \in [\mathsf{VC.Setup}(1^\lambda, S)]$, and $\boldsymbol{m} \in \mathcal{M}^S$, the following procedure always returns 1.

$$(C, (\mathsf{open}_i)_{i\in[0,S-1]}) \leftarrow\!\!\$\ \mathsf{VC.Com}(\mathsf{crs}, \boldsymbol{m})$$
$$\textbf{return } \forall i \in [0, S-1] : \mathsf{VC.V}(\mathsf{crs}, C, i, m_i, \mathsf{open}_i) = 1$$

Position-binding. No (efficient) adversary can output $(C, j, m_j, m'_j, \mathsf{open}_j, \mathsf{open}'_j)$ such that $(C, j, m_j, \mathsf{open}_j)$ and $(C, j, m'_j, \mathsf{open}_j)$ verify, but $m_j \neq m'_j$. More formally, for $S = S(\lambda)$, the advantage of any adversary $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC},S}(\mathcal{A}, \lambda) := \mathsf{Pr}\left[\begin{array}{l|l} m_i \neq m'_i \ \wedge\ i \in [0, S-1] & \mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, S) \\ \mathsf{VC.V}(\mathsf{crs}, C, i, m_i, \mathsf{open}_i) = 1 \ \wedge & (C, i, m_i, \mathsf{open}_i, m'_i, \mathsf{open}'_i) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}_{\mathsf{VC}}) \\ \mathsf{VC.V}(\mathsf{crs}, C, i, m'_i, \mathsf{open}'_i) = 1 & \end{array}\right].$$

Statistically hiding. For any *unbounded* adversary $\mathcal{A}$, the following advantage is bounded.

$$\mathsf{Adv}^{\mathsf{hide}}_{\mathsf{VC},S}(\mathcal{A}, \lambda) := \left|\mathsf{Pr}\left[b' = 1 \middle| \begin{array}{l} \mathsf{crs} \leftarrow\!\!\$\ \mathsf{VC.Setup}(1^\lambda, S) \\ (\boldsymbol{m}_0, \boldsymbol{m}_1, \mathsf{st}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}) \\ (C, (\mathsf{open}_i)_{i\in[0,S-1]}) \\ \qquad \leftarrow\!\!\$\ \mathsf{VC.Com}(\mathsf{crs}, \boldsymbol{m}_0) \\ b' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{st}, C) \end{array}\right] - \mathsf{Pr}\left[b' = 1 \middle| \begin{array}{l} \mathsf{crs} \leftarrow\!\!\$\ \mathsf{VC.Setup}(1^\lambda, S) \\ (\boldsymbol{m}_0, \boldsymbol{m}_1, \mathsf{st}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}) \\ (C, (\mathsf{open}_i)_{i\in[0,S-1]}) \\ \qquad \leftarrow\!\!\$\ \mathsf{VC.Com}(\mathsf{crs}, \boldsymbol{m}_1) \\ b' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{st}, C) \end{array}\right]\right|.$$

Succinctness. The size of each opening open_i and the verification time is at most $\mathsf{poly}(\lambda, \log S)$.

STATISTICAL PSEUDORANDOM FUNCTIONS. For a function family $\mathcal{F}_\lambda := \{F_{\mathsf{key}} : \mathcal{D}_\lambda \to \mathcal{R}_\lambda\}_{\mathsf{key}\in\mathcal{K}_\lambda}$ for $\lambda \in \mathbb{N}$, we say that the function is statistically pseudorandom, if the following advantage of any *unbounded adversary* $\mathcal{A}$ is bounded:

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{A}, \lambda) := |\mathsf{Pr}[\mathcal{A}^{\mathcal{O}_0}(1^\lambda) = 1 | \mathsf{key} \leftarrow\!\!\$\ \mathcal{K}_\lambda] - \mathsf{Pr}[\mathcal{A}^{\mathcal{O}_1}(1^\lambda) = 1 | \mathsf{key} \leftarrow\!\!\$\ \mathcal{K}_\lambda]|\ ,$$

where $\mathcal{O}_0(x)$ returns $F_{\mathsf{key}}(x)$, while $\mathcal{O}_1(x)$ keeps track of a table T, samples $T[x] \leftarrow\!\!\$\ \mathcal{R}$ if $T[x]$ is not initialized, and returns $T[x]$.

GENERALIZED FORKING LEMMA. We recall the generalized multi-forking lemma stated by Bagherzandi, Cheon, and Jarecki [BCJ08]. We will later use this lemma to show multi-proof rewinding extractability of a Fiat-Shamir compiled NIZK from sigma protocols.

Lemma 2.2 (Multi-Forking Lemma [BCJ08]). *Let $q \geqslant 1$ be an integer. Let $\mathcal{A}$ be a probabilistic algorithm which takes as input a main input inp generated by some probabilistic algorithm $\mathsf{IG}()$, elements $h_1, \ldots, h_q$ from some sampleable set H, and random coins ρ from some sampleable set $\mathcal{R}_{\mathcal{A}}$, and returns either a distinguished failure symbol $\perp$, or a tuple $(F, \{\phi_j\}_{j\in F}, \mathsf{aux})$, where $F \subseteq [q]$ and $F \neq \varnothing$, and $\{\phi_j\}_{j\in F}$ and aux are some auxiliary outputs. The accepting probability of $\mathcal{A}$ denoted acc is defined as the probability (over $\mathsf{inp}, h_1, \ldots, h_q, \rho$) that $\mathcal{A}$ returns a non-$\perp$ output. Consider the following algorithm $\mathsf{MFork}^{\mathcal{A}}(\mathsf{inp})$:*

- $\rho \leftarrow\!\!\$\ R_{\mathcal{A}}, h_1, \ldots, h_q \leftarrow\!\!\$\ H$
- *Run $\mathcal{A}(\mathsf{inp}, h_1, \ldots, h_q; \rho)$ and abort if $\mathcal{A}$ return $\perp$. Otherwise, parse $\mathcal{A}$'s output as $(F, \{\phi_j\}_{j\in F}, \mathsf{aux})$.*

- *Set* $\mathsf{out} \leftarrow \{(h_j, \phi_j)\}_{j \in F}$ *and* $\mathsf{out}' \leftarrow \varnothing$.
- *For each* $j \in F$*:*
  - *Set* $\mathsf{succ} \leftarrow \mathtt{false}, K \leftarrow 0, K_{max} \leftarrow \frac{8|F|q}{\mathsf{acc}} \ln(\frac{8|F|}{\mathsf{acc}})$.
  - *Repeat until* $\mathsf{succ} = \mathtt{true}$ *or* $K \geqslant K_{max}$
    1. $K \leftarrow K + 1; h'_j, \ldots, h'_q \leftarrow\!\!\$\ H$
    2. *Run* $(F', \{\phi'_j\}_{j \in F}, \mathsf{aux}') \leftarrow \mathcal{A}(\mathsf{inp}, h_1, \ldots, h_{j-1}, h'_j, \ldots, h'_q; \rho)$.
    3. *If* $F' \neq \perp$ *and* $j \in F'$ *and* $h'_j \neq h_j$ *then set* $\mathsf{succ} \leftarrow \mathtt{true}$ *and* $\mathsf{out}' \leftarrow \mathsf{out}' \cup \{(h'_j, \phi'_j)\}$
  - *If* $\mathsf{succ} = \mathtt{false}$ *then abort.*
- *Return* $(F, \mathsf{out}, \mathsf{out}', \mathsf{aux})$.

Let mfrk *denote the probability (over* inp *and its random coins) that* MFork *does not abort. Then, assume* $|H| > 8|F|q/\mathsf{acc}$*, then* $\mathsf{mfrk} \geqslant \frac{\mathsf{acc}}{8}$.

3 Everlasting Anonymous Rate-Limited Tokens (EARLT)

We introduce the syntax of an Everlasting Anonymous Rate-Limited Tokens (EARLT) modified from the syntax and definition of "Periodic Anonymous Authentication" given in [CHK+06].

Syntax. Everlasting Anonymous Rate-Limited Tokens allow users to obtain stateful token dispensers from interacting with an issuer I. Each dispenser can create up to $N = N(\lambda)$ unlinkable tokens per each context (denoted uniquely by ctxt) and at most $k = k(\lambda)$ tokens overall. The users can then show these tokens to a verifier to authenticate themselves. The scheme consists of the following algorithms, which has RAM access (except for Setup) to the CRS written in its memory (i.e., we assume accessing values in the CRS takes constant time).

- $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, k, N)$. The setup algorithm generates the common-reference string crs which also defines a space of contexts $\mathcal{C} = \mathcal{C}_{\mathsf{crs}}$ and serial numbers $\mathcal{S} = \mathcal{S}_{\mathsf{crs}}$ (usually denoted with the subscript omitted).
- $(\mathsf{sk}_\mathsf{I}, \mathsf{pk}_\mathsf{I}) \leftarrow\!\!\$\ \mathsf{IKGen}(\mathsf{crs})$. The issuer key generation algorithm generates the secret and public keys of the issuer.
- $(\mathsf{sk}_{\mathsf{User}}, \mathsf{pk}_{\mathsf{User}}) \leftarrow\!\!\$\ \mathsf{UKGen}(\mathsf{crs})$. The user key generation algorithm generates the secret and public keys of the user.
- $(\perp, \mathsf{D}) \leftarrow\!\!\$\ \langle \mathsf{I}(\mathsf{crs}, \mathsf{sk}_\mathsf{I}, \mathsf{pk}_{\mathsf{User}}) \rightleftharpoons \mathsf{User}(\mathsf{crs}, \mathsf{sk}_{\mathsf{User}}, \mathsf{pk}_\mathsf{I}) \rangle$. The issuance protocol allows the user to obtain a token dispenser D by interacting with the issuer. We consider a round-optimal issuance protocol with the following structure:
  - $(\mathsf{st}^u, \mathsf{umsg}) \leftarrow\!\!\$\ \mathsf{User}_1(\mathsf{sk}_{\mathsf{User}}, \mathsf{pk}_\mathsf{I})$
  - $\mathsf{imsg} \leftarrow\!\!\$\ \mathsf{I}(\mathsf{sk}_\mathsf{I}, \mathsf{pk}_{\mathsf{User}}, \mathsf{umsg})$
  - $\mathsf{D} \leftarrow\!\!\$\ \mathsf{User}_2(\mathsf{st}^u, \mathsf{imsg})$
- Showing and verification: A protocol where the user shows a token generated for a context string $\mathsf{ctxt} \in \mathcal{C}$ to the verifier V. In our case, we consider protocols with the following structure:
  - The verifier computes a random nonce $R \leftarrow\!\!\$\ \mathsf{nonce}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt})$.
  - The user runs the showing algorithm $((\mathsf{sn}, \tau), \mathsf{D}') \leftarrow\!\!\$\ \mathsf{Show}(\mathsf{D}, \mathsf{ctxt}, R)$, with RAM access to the dispenser D initialized in its memory, and sends the token τ and the serial number sn to the verifier.
  - The verifier checks (sn, τ) via $b \leftarrow \mathsf{V}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt}, R, \mathsf{sn}, \tau)$.
- $\mathsf{pk}'_{\mathsf{User}} \leftarrow \mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt}, R, R', \mathsf{sn}, \tau, \tau')$. The identification algorithm returns the identity/public key of a user who double-spends a token.

Correctness. An EARLT scheme is correct if for any $\lambda \in \mathbb{N}, N = N(\lambda), k = k(\lambda)$, any $\mathsf{crs} \in [\mathsf{Setup}(1^\lambda, k, N)]$ and any sequence of context strings $\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_k \in \mathcal{C}$ such that *no context string repeats more than* N *times*,

the following experiment always returns 1.

$$(\mathsf{sk}_\mathsf{I}, \mathsf{pk}_\mathsf{I}) \leftarrow\!\!\$\ \mathsf{IKGen}(\mathsf{crs})\ ;\ (\mathsf{sk}_\mathsf{User}, \mathsf{pk}_\mathsf{User}) \leftarrow\!\!\$\ \mathsf{UKGen}(\mathsf{crs})$$
$$(\bot, \mathsf{D}_0) \leftarrow\!\!\$\ \langle \mathsf{I}(\mathsf{sk}_\mathsf{I}, \mathsf{pk}_\mathsf{User}) \rightleftharpoons \mathsf{User}(\mathsf{sk}_\mathsf{User}, \mathsf{pk}_\mathsf{I})\rangle$$
$$\text{For } i \in [k]\text{: } R_i \leftarrow\!\!\$\ \mathsf{nonce}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt}_i)$$
$$\text{For } i \in [k]\text{: } ((\mathsf{sn}_i, \tau_i), \mathsf{D}_i) \leftarrow\!\!\$\ \mathsf{Show}(\mathsf{D}_{i-1}, \mathsf{ctxt}_i, R_i)$$
$$\textbf{return } (\forall i \in [k] : \mathsf{V}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt}_i, R_i, \mathsf{sn}_i, \tau_i) = 1)$$

$(t_\mathsf{User}, t_\mathsf{I}, t_\mathsf{Show}, t_\mathsf{V}, s_\tau)$-EFFICIENCY. The specified functions $t_\mathsf{User}(\lambda, N, k)$, $t_\mathsf{I}(\lambda, N, k)$, $t_\mathsf{Show}(\lambda, N, k)$, $t_\mathsf{V}(\lambda, N, k)$ and $s_\tau(\lambda, N, k)$ are such that $t_\mathsf{I}, t_\mathsf{Show}, t_\mathsf{V}, s_\tau$ are sublinear in N and k (i.e., $o(N + k) \cdot \mathsf{poly}(\lambda)$), while t_User is quasilinear in N and k (i.e., $\mathsf{poly}(\lambda, \log N, \log k)$). We then have the following guarantees:

- The user algorithm User runs in time $t_\mathsf{User}(\lambda, N, k)$.
- The issuer algorithm I runs in time $t_\mathsf{I}(\lambda, N, k)$.
- The showing algorithm Show runs in time $t_\mathsf{Show}(\lambda, N, k)$.
- The verification algorithm V runs in time $t_\mathsf{V}(\lambda, N, k)$.
- The size of the token τ is $s_\tau(\lambda, N, k)$.

SECURITY DEFINITIONS. For our scheme, we consider four security notions: everlasting anonymity, unforgeability, identification, and exculpability.

Everlasting anonymity. The anonymity game is defined in a similar manner to [CHK+06], except that we introduce an additional condition that a token dispenser cannot produce more than k tokens (where k is explicitly parametrizing the game). Note that this is essential in achieving everlasting anonymity simply because we cannot issue a dispenser which is bound to unbounded randomness source. In the anonymity game ANON, formally described in Figure 4, the goal of the adversary is to guess whether it is interacting with honest users or a simulator $\mathsf{Sim} = (\mathsf{Sim}_\mathsf{Setup}, \mathsf{Sim}_\mathrm{U} = (\mathsf{Sim}_1, \mathsf{Sim}_2), \mathsf{Sim}_\mathsf{Show})$. The adversary has access to the following oracles:

- INIT allows the adversary to establish a (possibly) malicious public key.
- NEWUSR allows the adversary to register a new honest user and obtain this user's public key.
- $\mathrm{U}_1, \mathrm{U}_2$ allow the adversary to issue tokens to honest users of its choice.
- SHOW allows the adversary to request showing tokens for a particular context string ctxt from a chosen token dispenser tied to an ID cid. Each dispenser only shows at most k tokens overall and N tokens per context.

The anonymity advantage of an adversary $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{EARLT},k,N,\mathsf{Sim}}(\mathcal{A}, \lambda) := |\mathsf{Pr}[\mathrm{ANON}^{\mathcal{A}}_{\mathsf{EARLT},k,N,\mathsf{Sim},0}(\lambda) = 1] - \mathsf{Pr}[\mathrm{ANON}^{\mathcal{A}}_{\mathsf{EARLT},k,N,\mathsf{Sim},1}(\lambda) = 1]| \ .$$

Unforgeability. Our unforgeability notion has similar nature to blind signatures' one-more unforgeability. It ensures that no adversary can show more than N tokens for a context string per each issued dispenser. Formally, this is described in the game UNF in Figure 5 where the adversary has access to an issuance oracle which it queries for Q times, and its goal is to output $Q \cdot N + 1$ valid tokens with different serial numbers corresponding to the same context string. The corresponding advantage of the adversary $\mathcal{A}$ is defined as

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

The prior work [CHK+06] defined a similar security notion, denoted soundness, which requires extracting in each issuance protocol an underlying function $f_i : \mathcal{C} \times \mathcal{I} \to \mathcal{S}$ which the user will use to compute the serial number. In their case, the adversary will win if it outputs a valid token τ with serial number sn for a context ctxt such that $\mathsf{sn} \neq f_i(\mathsf{ctxt}, x)$ for any input $x \in \mathcal{I}$. Looking ahead, since the keys of our function will be large (the size scales with k), we will need expensive online extractable proofs of knowledge for it.

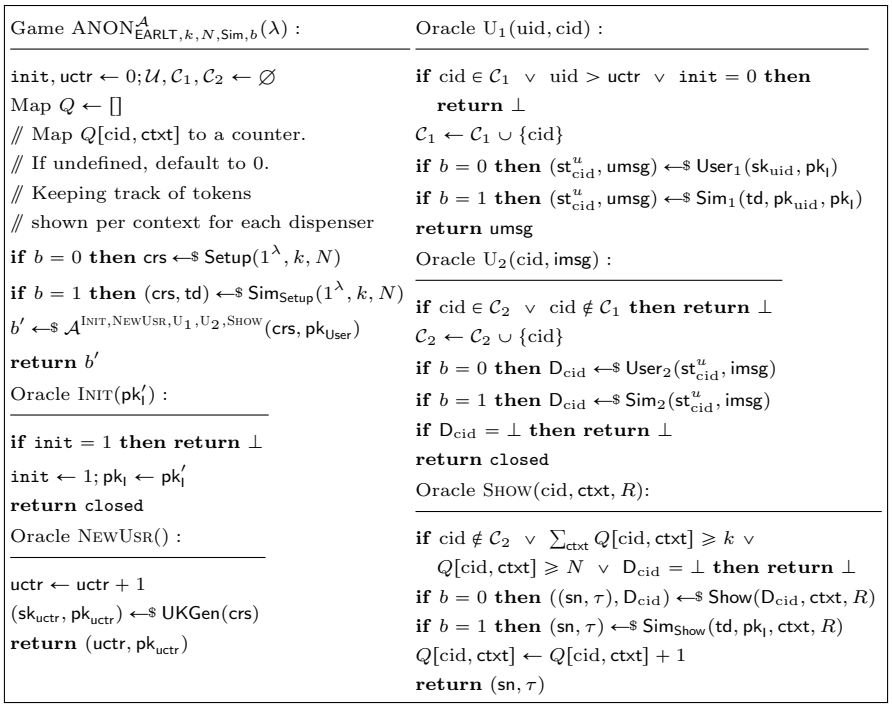

Fig. 4. Anonymity Game.

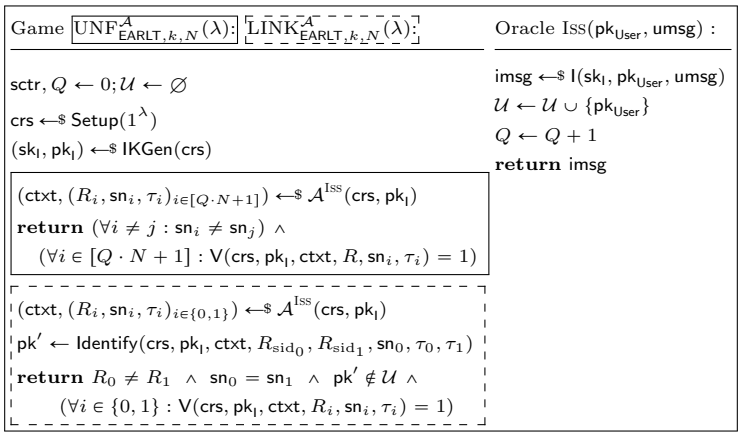

Fig. 5. Unforgeability and Linkability Game. The codes in solid and dashed boxes are exclusive to the unforgeability and linkability games, resp.

Therefore, we only consider the weaker one-more unforgeability notion, which is also simpler to understand and capture the rate-limiting property of the tokens.

Linkability. Linkability aims to provide accountability against the users who try to double-spend their tokens. In particular, if the adversary outputs two valid tokens τ_0, τ_1 for a particular context ctxt with the same serial number $\mathsf{sn}_0 = \mathsf{sn}_1$, then the verifier should be able to link the double-spent tokens to a user public key. More formally, the linkability game (defined via the LINK game in Figure 5) prevents an adversary $\mathcal{A}$ (with access to issuing oracle) from outputting 2 tuples of tokens and nonces $(R_0, \mathsf{sn}_0, \tau_0), (R_1, \mathsf{sn}_1, \tau_1)$ for the same context string ctxt with the same serial number $\mathsf{sn}_0 = \mathsf{sn}_1$ and different nonces $R_0 \neq R_1$, but $\mathsf{Identify}$ does not return any public key $\mathsf{pk}_{\mathsf{User}}$ of a user with a dispenser issued. The advantage of the adversary $\mathcal{A}$ in

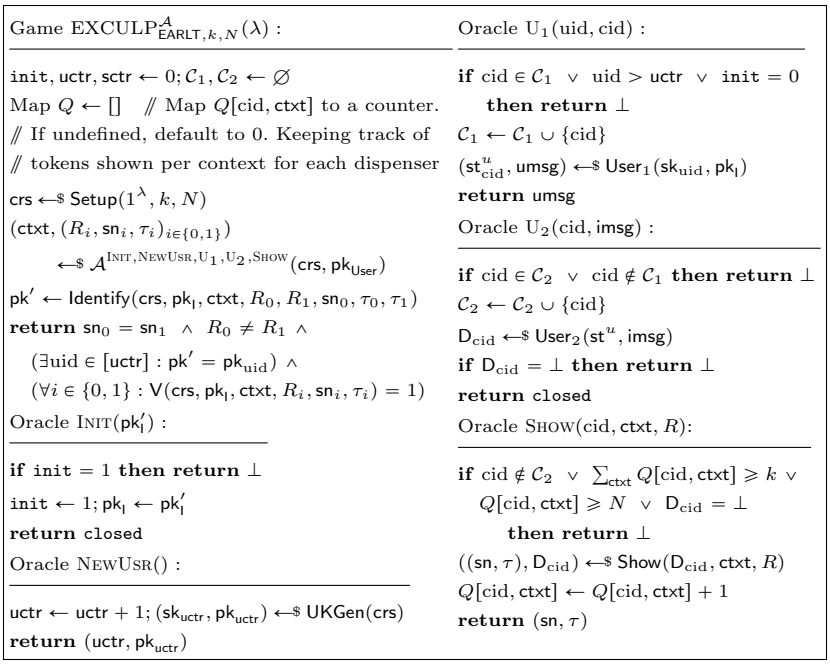

Fig. 6. Exculpability Game.

this game is

$$\mathsf{Adv}^{\mathsf{link}}_{\mathsf{EARLT},k,N}(\mathcal{A}, \lambda) := \mathsf{Pr}[\text{LINK}^{\mathcal{A}}_{\mathsf{EARLT},k,N}(\lambda) = 1] \ .$$

We note that this property is similar to the identification property defined in [CHK+06], except we allow multiple malicious users (via multiple issuance oracle call) *instead of one*malicious user, and we only require the nonces to be distinct, instead of honestly generated.

Remark 3.1. Our unforgeability and linkability do not provide rate-limiting gua-rantee with respect to k. More precisely, our unforgeability does not guarantee that "no adversary can produce $kQ+1$ valid tokens with distinct serial numbers (but *without* restrictions on the verified context ctxt being the same for all tokens)". Analogously, our linkability does not guarantee that "no adversary can produce more than k tokens (where some of them might contain the same serial number) and not be identified". We discuss later on in Section 5 a simple modification to our construction to achieve these alternative sketched notions.

Also, if one does not care about rate-limiting per context, we can set the context class $\mathcal{C}$ to contains only one string, e.g., an all-zero string 0^m for some m and $N = k$ to achieve the rate-limiting guarantee discussed above.

Exculpability. In addition to linkability, we require that no group of adversarial users and *a malicious issuer* can frame honest users for double-spending. This is modeled as the EXCULP game (in Figure 6) where the adversary can ask to generate new honest users (via NEWUSR), ask these users to request token generators (via U_1, U_2 with the adversary acting as a malicious issuer), and request the honest users to show tokens within some context string (via SHOW). These oracles are similar to those of the anonymity game. The goal of the adversary is to output two tokens with the same serial number that identifies to the public key of an honest user. The advantage of $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{exculp}}_{\mathsf{EARLT},k,N}(\mathcal{A}, \lambda) := \mathsf{Pr}[\text{EXCULP}^{\mathcal{A}}_{\mathsf{EARLT},k,N}(\lambda) = 1] \ .$$

Game $\mathrm{UNF}^{\mathcal{A}}_{\mathsf{SPS}}(\lambda)$:	Oracle $\mathrm{S}(\boldsymbol{M})$:
$Q \leftarrow \varnothing$	$Q \leftarrow Q \cup \{\boldsymbol{M}\}$
$\mathsf{par} \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda)$; $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{SPS.KeyGen}(\mathsf{par})$	$\sigma \leftarrow\!\!\$\ \mathsf{SPS.S}(\mathsf{sk}, \boldsymbol{M})$
$(\boldsymbol{M}^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{S}}(\mathsf{par}, \mathsf{pk})$	**return** σ
return $\boldsymbol{M}^* \notin Q \ \wedge\ \mathsf{SPS.V}(\mathsf{pk}, \boldsymbol{M}, \sigma) = 1$	

Fig. 7. Unforgeability game for the scheme $\mathsf{SPS} = \mathsf{SPS}[\mathsf{GGen}]$.

4 Building Blocks

In this section, we give definitions and also scheme descriptions for the building blocks of our construction. All of the building blocks will depends on the bilinear pairing groups parameters generated as $\mathsf{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda)$.

Structure-preserving signatures. A structure-preserving signature [AFG+10] sch-eme $\mathsf{SPS} = \mathsf{SPS}[\mathsf{GGen}] = (\mathsf{SPS.KeyGen}, \mathsf{SPS.S}, \mathsf{SPS.V})$has its public key, messages, and signatures being collections of source group elements (i.e., $\mathbb{G}_1, \mathbb{G}_2$ elements), with the following syntax:

- Key generation: $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{SPS.KeyGen}(\mathsf{par})$.
- Signing algorithm: $\sigma \leftarrow\!\!\$\ \mathsf{SPS.S}(\mathsf{sk}, \boldsymbol{M} \in \mathbb{G}_1^n)$.
- $0/1 \leftarrow \mathsf{SPS.V}(\mathsf{pk}, \boldsymbol{M}, \sigma)$. The verification can be expressed solely as pairing-product equations involving group elements contained in pk, σ and $\boldsymbol{M}$.

Note that we give the syntax for unilateral messages (i.e., messages in $\mathbb{G}_1^n$and not $\mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2}$) which are what our scheme requires. We require SPS to be correct and existentially unforgeable (EUF-CMA) as defined below. Throughout the work, we will use SPS as a black-box.

Correctness. For any $\mathsf{par} \in [\mathsf{GGen}(1^\lambda)]$, $\boldsymbol{M} \in \mathbb{G}_1^{n_1}$, the following procedure always return 1.

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{SPS.KeyGen}(\mathsf{par})\ ;$$
$$\sigma \leftarrow\!\!\$\ \mathsf{SPS.S}(\mathsf{sk}, \boldsymbol{M})$$
$$\textbf{return}\ \mathsf{SPS.V}(\mathsf{pk}, \boldsymbol{M}, \sigma)\ .$$

Unforgeability. No efficient adversary with access to a signing oracle can forge a signature on a message that was not previously queried to the oracle. We denote the advantage of any adversary $\mathcal{A}$ playing the UNF defined in Figure 7 as

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

We remark that there exists several constructions [AHN+23, GHKP18, AJOR18, AJO+19] for structure-preserving signatures from the standard SXDH assumption with *constant size* signature and verification time scaling linearly with the number of group elements in the message.

KZG commitments. We state the constructions of two variants of KZG commitment schemes [KZG10], $\mathsf{KZG} = \mathsf{KZG}[\mathsf{GGen}]$ and $\mathsf{KZG}_{\mathsf{Ped}} = \mathsf{KZG}_{\mathsf{Ped}}[\mathsf{GGen}]$, in Figure 8. The scheme KZG is correct, evaluation-binding under d-SDH assumption, and degree-binding under d-ARSDH assumption. The scheme $\mathsf{KZG}_{\mathsf{Ped}}$ is additionally *perfectly hiding*. We show these property in the full version.

KZG commitments are correct and evaluation-binding under d-SDH assumption as established in the following lemma.

Lemma 4.1 ([KZG10]). *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order $p = p(\lambda)$ and $d = d(\lambda) \in \mathbb{N}$. The polynomial commitment scheme $\mathsf{KZG} = \mathsf{KZG}[\mathsf{GGen}]$ commitment is correct, perfectly hiding, evaluation-binding under d-SDH assumption. In particular, for any adversary $\mathcal{A}$ running in time $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$, there exists an adversary $\mathcal{B}$ running in time roughly $t_{\mathcal{A}}$ such that*

$$\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG},d}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{d\text{-SDH}}_{\mathsf{GGen}}(\mathcal{B}, \lambda)\ .$$

Algorithm $\mathsf{KZG.Setup}(\mathsf{par}, d)$:

$x \leftarrow\!\!\$\ \mathbb{Z}_p$

return $((X_{1,i} = x^i G_1)_{i\in[d]}, X_{2,1} = xG_2)$

Algorithm $\mathsf{KZG.Com}(\mathsf{crs}, f \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}])$:

return $(C = \sum_{i=0}^{d} f_i X_{1,i}, \rho = \bot)$

Algorithm $\mathsf{KZG.Open}(\mathsf{crs}, C, f, \alpha, \beta, \rho)$:

return $Q = \frac{f(x)-\beta}{(x-\alpha)} G_1$

Algorithm $\mathsf{KZG.V}(\mathsf{crs}, C, \alpha, \beta, Q)$:

return $\mathsf{e}(Q, X_{2,1} - \alpha G_2) = \mathsf{e}(C - \beta G_1, G_2)$

Algorithm $\mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Setup}(\mathsf{par}, d)$:

$x \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ H \leftarrow\!\!\$\ \mathbb{G}_1^*$

for $i \in [d] : X_{1,i} \leftarrow x^i G_1\ ; \hat{X}_{1,i} \leftarrow x^i H$

return $(H, (X_{1,i}, \hat{X}_{1,i})_{i\in[d]}, X_{2,1} = xG_2)$

Algorithm $\mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Com}(\mathsf{crs}, f \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}])$:

$g \leftarrow\!\!\$\ \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$

return $(C = \sum_{i=0}^{d} f_i X_{1,i} + g_i \hat{X}_{1,i}, \rho = g)$

Algorithm $\mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Open}(\mathsf{crs}, C, f, \alpha, \beta, \rho)$:

$\beta' \leftarrow g(\alpha)$

return $(\beta', Q = \frac{f(x)-\beta}{(x-\alpha)} G_1 + \frac{g(x)-\beta'}{(x-\alpha)} H)$

Algorithm $\mathsf{KZG}_{\mathsf{Ped}}.\mathsf{V}(\mathsf{crs}, C, \alpha, \beta, (\beta', Q))$:

return $\mathsf{e}(Q, X_{2,1} - \alpha G_2) = \mathsf{e}(C - \beta G_1 - \beta' H, G_2)$

Algorithm $\mathsf{VC}_{\mathsf{KZG}}.\mathsf{Setup}(\mathsf{par}, d)$:

return $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{KZG.Setup}(\mathsf{par}, d)$

Algorithm $\mathsf{VC}_{\mathsf{KZG}}.\mathsf{V}(\mathsf{crs}, C, i, m_i, Q)$:

return $\mathsf{KZG.V}(\mathsf{crs}, C, \alpha, \beta, Q)$

Algorithm $\mathsf{VC}_{\mathsf{KZG}}.\mathsf{Com}(\mathsf{crs}, \boldsymbol{m} \in \mathbb{Z}_p^S)$:

Let $f \in \mathbb{Z}_p^{\leqslant S}[\mathsf{X}]$ s.t. $\forall i \in [0, S-1] : f(i) = m_i$

$\quad \wedge\ f(S) = r$ for $r \leftarrow\!\!\$\ \mathbb{Z}_p$

$C \leftarrow \mathsf{KZG.Com}(\mathsf{crs}, f)$

for $i \in [0, S-1] : \mathsf{open}_i \leftarrow \mathsf{KZG.Open}(\mathsf{crs}, C, f, i, m_i)$

return $(C, (\mathsf{open}_i)_{i\in[0,S-1]})$

Fig. 8. Variants of KZG commitment scheme $\mathsf{KZG}, \mathsf{KZG}_{\mathsf{Ped}}$, and $\mathsf{VC}_{\mathsf{KZG}}$. Public parameters par is generated from GGen.

The following lemma establishes the security of $\mathsf{KZG}_{\mathsf{Ped}}$ commitment scheme. Correctness, perfectly hiding, and evaluation-binding follows from the results in [KZG10]. For the degree-binding property, we refer to Section B for the security proof.

Lemma 4.2. *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order* $p = p(\lambda)$ *and* $d = d(\lambda) \in \mathbb{N}$. *The polynomial commitment scheme* $\mathsf{KZG}_{\mathsf{Ped}} = \mathsf{KZG}_{\mathsf{Ped}}[\mathsf{GGen}]$ *commitment is correct, perfectly hiding, evaluation-binding under* d*-SDH assumption, and degree-binding under* d-ARSDH *assumption. In particular, for any adversary* $\mathcal{A}$ *running in time* $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$, *there exist adversaries* $\mathcal{B}, \mathcal{B}'$, *running in time roughly* $t_{\mathcal{A}}$ *such that*

$$\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{d\text{-SDH}}_{\mathsf{GGen}}(\mathcal{B}, \lambda)\ ,\ \mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{A}, \lambda) \leqslant 3 \cdot \mathsf{Adv}^{d\text{-ARSDH}}_{\mathsf{GGen}}(\mathcal{B}', \lambda)\ .$$

VECTOR COMMITMENT FROM KZG. We also use vector commitment scheme derived from KZG, which has been sketched before in [KZG10]. The vector commitment scheme $\mathsf{VC}_{\mathsf{KZG}} = \mathsf{VC}_{\mathsf{KZG}}[\mathsf{GGen}]$ is also defined in Figure 8. For groups of order p such that $(p-1)$ is divisible by $2S = 2^{m+1}$, the openings to all positions $i \in [0, S-1]$ can be computed all at once in $O(S \log^2 S)$ group exponentiations [FK23] (a similar statement can be shown from an observation in [GHO20]). The following lemma, proved in Section B.2, establishes the security of $\mathsf{VC}_{\mathsf{KZG}}$.

Lemma 4.3. *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order* $p = p(\lambda)$ *and* $S = S(\lambda) \in \mathbb{N}$. *The polynomial commitment scheme* $\mathsf{VC}_{\mathsf{KZG}} = \mathsf{VC}_{\mathsf{KZG}}[\mathsf{GGen}]$ *commitment is correct, statistically hiding, position-binding under* S*-SDH assumption, and succinct. In particular, for any unbounded adversary* $\mathcal{A}_{\mathsf{hide}}$ *and adversary* $\mathcal{A}_{\mathsf{pbind}}$ *running in time* $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$, *there exists an adversary* $\mathcal{B}$ *running in time roughly* $t_{\mathcal{A}}$ *such that*

$$\mathsf{Adv}^{\mathsf{hide}}_{\mathsf{VC}_{\mathsf{KZG}},S}(\mathcal{A}_{\mathsf{hide}}, \lambda) \leqslant \frac{S}{p}\ , \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},S}(\mathcal{A}_{\mathsf{pbind}}, \lambda) \leqslant \mathsf{Adv}^{S\text{-SDH}}_{\mathsf{GGen}}(\mathcal{B}, \lambda)\ .$$

GROTH-SAHAI NON-INTERACTIVE PROOFS. We recall the syntax and security properties of Groth-Sahai [GS08] (GS) non-interactive proof system. We follow the formalization of the proof system as a commit-and-proof scheme as given in [EG14]. The GS proof system is a non-interactive zero-knowledge proof for satisfiability of a set of equations over pairing-based group described by par. Let $\mathcal{R}_{\mathsf{par}}$ denote a family of relations over $\mathsf{par} \in [\mathsf{GGen}(1^\lambda)]$ and for a relation $\mathcal{R} \in \mathcal{R}_{\mathsf{par}}$, let $\mathcal{L}_{\mathcal{R}}$ be the corresponding induced language. In the commit-and-prove scheme, we may commit to different witnesses $w_1, \ldots, w_L$ where each of them is of the form (t_i, m_i) where t_i denotes the public type of the committed value (i.e., $\mathbb{G}_1$ or $\mathbb{G}_2$ element or scalars committed in $\mathbb{G}_1$ or $\mathbb{G}_2$) and m_i is the committed value. We will omit the public type for readability. The Groth-Sahai commit-and-prove system $\mathsf{GS} = \mathsf{GS}[\mathsf{GGen}]$ with the setup algorithms taking as input $\mathsf{par} \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda)$ and $\mathcal{R} \in \mathcal{R}_{\mathsf{par}}$consists of the following algorithms:

- $\mathsf{crs}_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{GS.Setup}_{\mathsf{bind}}(\mathsf{par})$ generates the CRS in binding mode.
- $\mathsf{crs}_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{GS.Setup}_{\mathsf{hide}}(\mathsf{par})$ generates the CRS in hiding mode.
- $\mathsf{com} \leftarrow\!\!\$\ \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \mathbb{w}; \mathsf{rand})$ or we write $(\mathsf{com}, \mathsf{rand}) \leftarrow\!\!\$\ \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \mathbb{w})$ with $\mathsf{rand} \in \mathsf{Rand}_{\mathsf{par}} = \mathbb{Z}_p^2$ denoting the sampled randomness. We will often abuse the notation with $\mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \mathbb{w}; \mathsf{rand})$ for $\mathbb{w}$ of more than one element, in which case, the randomness size scales with the number of elements.
- $\pi \leftarrow\!\!\$\ \mathsf{GS.P}(\mathsf{crs}_{\mathsf{GS}}, (\mathbb{x}, \mathsf{com}), (\mathbb{w}, \mathsf{rand}))$ computes a proof π for a statement $\mathbb{x}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\mathsf{com} = \mathsf{Com}(\mathsf{crs}_{\mathsf{GS}}, \mathbb{w}; \mathsf{rand})$.
- $0/1 \leftarrow \mathsf{GS.V}(\mathsf{crs}_{\mathsf{GS}}, (\mathbb{x}, \mathsf{com}), \pi)$

Note that the relation $\mathcal{R}$ contains statements and witnesses $(\mathbb{x}, \mathbb{w})$ corresponding to the following types of equations[5]:

Pairing-product equation. Consider public constants in the statement $\mathbb{x}$: $A_j \in \mathbb{G}_1, B_i \in \mathbb{G}_2, \gamma_{i,j} \in \mathbb{Z}_p$ and $T_{\mathbb{G}_T} \in \mathbb{G}_T$ and the witness containing $X_i \in \mathbb{G}_1, Y_j \in \mathbb{G}_2$ such that

$$\sum_i \mathsf{e}(X_i, B_i) + \sum_j \mathsf{e}(A_j, Y_j) + \sum_{i,j} \gamma_{i,j} \mathsf{e}(X_i, Y_j) = T_{\mathbb{G}_T} .$$

Multi-scalar multiplication in $\mathbb{G}_1$ or $\mathbb{G}_2$. We state for $\mathbb{G}_1$ and $\mathbb{G}_2$ is analogous. Consider public constants in the statement $\mathbb{x}$: $A_j \in \mathbb{G}_1, b_i \in \mathbb{Z}_p, \gamma_{i,j} \in \mathbb{Z}_p$ and $T_1 \in \mathbb{G}_1$ and the witness containing $X_i \in \mathbb{G}_1, y_j \in \mathbb{Z}_p$ such that

$$\sum_i b_i X_i + \sum_j y_j A_j + \sum_{i,j} \gamma_{i,j} y_j X_i = T_1 .$$

Quadratic equation in $\mathbb{Z}_p$. Consider public constants in the statement $\mathbb{x}$: $a_j, b_i \in \mathbb{Z}_p, \gamma_{i,j} \in \mathbb{Z}_p$ and $t \in \mathbb{Z}_p$ and the witness containing $x_i, y_j \in \mathbb{Z}_p$ such that

$$\sum_i b_i x_i + \sum_j y_j a_j + \sum_{i,j} \gamma_{i,j} y_j x_i = t .$$

We require the following properties of Groth-Sahai proof to hold for all $\mathsf{par} \in [\mathsf{GGen}(1^\lambda)]$:

Perfect correctness. For any $\mathcal{R} \in \mathcal{R}_{\mathsf{par}}$ and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the following procedure always return 1.

$$\begin{aligned}
&\mathsf{crs}_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{GS.Setup}_{\mathsf{bind/hide}}(\mathsf{par}); (\mathsf{com}, \mathsf{rand}) \leftarrow\!\!\$\ \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \mathbb{w}) \\
&\pi \leftarrow\!\!\$\ \mathsf{GS.P}(\mathsf{crs}_{\mathsf{GS}}, (\mathbb{x}, \mathsf{com}), (\mathbb{w}, \mathsf{rand})) \\
&\textbf{return } \mathsf{GS.V}(\mathsf{crs}_{\mathsf{GS}}, (\mathbb{x}, \mathsf{com}), \pi) .
\end{aligned}$$

CRS indistinguishability. The distributions of $\mathsf{crs}_{\mathsf{GS}}$ generated from $\mathsf{Setup}_{\mathsf{bind}}$ and $\mathsf{Setup}_{\mathsf{hide}}$ are computationally indistinguishable, i.e., for any $\mathcal{A}$, the following advantage is bounded:

$$\mathsf{Adv}^{\mathsf{dist}}_{\mathsf{GS}}(\mathcal{A}, \lambda) := \left| \Pr\left[\mathcal{A}(\mathsf{crs}_{\mathsf{GS}}) = 1 \middle| \begin{matrix} \mathsf{par} \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda) \\ \mathsf{crs}_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{Setup}_{\mathsf{bind}}(\mathsf{par}) \end{matrix} \right] - \Pr\left[\mathcal{A}(\mathsf{crs}_{\mathsf{GS}}) = 1 \middle| \begin{matrix} \mathsf{par} \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda) \\ \mathsf{crs}_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{Setup}_{\mathsf{hide}}(\mathsf{par}) \end{matrix} \right] \right| .$$

In particular, there exists an adversary $\mathcal{B}$ playing the SXDH game such that $\mathsf{Adv}^{\mathsf{dist}}_{\mathsf{GS}}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{\mathsf{sxdh}}_{\mathsf{GGen}}(\mathcal{B}, \lambda)$.

[5] The verification of SPS and KZG commitments are of these formats.

Perfectly binding in binding mode. For all $\mathsf{crs}_{\mathsf{GS}} \in \mathsf{GS.Setup}_{\mathsf{bind}}(\mathsf{par})$, for any $a \neq b$ of the same type,

$$\{\mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, a; r) : r \in \mathbb{Z}_p^2\} \cap \{\mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, b; r) : r \in \mathbb{Z}_p^2\} = \varnothing \ .$$

Perfect soundness in binding mode. For all $\mathsf{crs}_{\mathsf{GS}} \in [\mathsf{GS.Setup}_{\mathsf{bind}}(\mathsf{par})]$, all $\mathbb{x} \notin \mathcal{L}_{\mathcal{R}}$, all commitments com, and all proofs π, $\mathsf{GS.V}(\mathsf{crs}_{\mathsf{GS}}, (\mathbb{x}, \mathsf{com}), \pi) = 0$.

Perfect F-Knowledge in binding mode. There exists efficient extractors $\mathsf{Ext}_{\mathsf{Setup}}, \mathsf{Ext}_{\mathsf{P}}$ such that

- $\mathsf{Ext}_{\mathsf{Setup}}(\mathsf{par})$: outputs $(\mathsf{crs}_{\mathsf{GS}}, \mathsf{td})$ such that $\mathsf{crs}_{\mathsf{GS}}$ is distributed identically to $\mathsf{Setup}_{\mathsf{bind}}(\mathsf{par})$
- $\mathsf{Ext}_{\mathsf{P}}(\mathsf{td}, \mathsf{com})$: outputs some value W.

Then, for any com, we require that there exists an opening $(\mathbb{w}, r)$ of com such that $W = F(\mathbb{w})$. Here, the function F checks if the each value in the witness $\mathbb{w}$ is a group element or scalar, for group elements, they are just returned as is, and for scalars, it returns a group element in the respective groups with the scalar as the discrete log with respect to the generator G_1 or G_2.

Perfect Zero-Knowledge in hiding mode. There exists efficient simulators $\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{\mathsf{Com}}, \mathsf{Sim}_{\mathsf{P}}$ such that

- $\mathsf{Sim}_{\mathsf{Setup}}(\mathsf{par})$: outputs $(\mathsf{crs}_{\mathsf{GS}}, \mathsf{td})$ such that $\mathsf{crs}_{\mathsf{GS}}$ is distributed identically to $\mathsf{Setup}_{\mathsf{hide}}(\mathsf{par})$.
- $\mathsf{Sim}_{\mathsf{Com}}(\mathsf{td})$: outputs a commitment com' and auxiliary value ρ.
- $\mathsf{Sim}_{\mathsf{P}}(\mathsf{td}, \mathbb{x}, \rho)$: outputs a proof π.

Then, for any $(\mathsf{crs}_{\mathsf{GS}}, \mathsf{td}) \in [\mathsf{Sim}_{\mathsf{Setup}}(\mathsf{par})]$ and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ the following two distributions are identical:

$$\left\{(\mathsf{com}, \pi) : \begin{array}{l} \mathsf{rand} \leftarrow\!\!\$\ \mathsf{Rand}_{\mathsf{par}}, \mathsf{com} \leftarrow \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \mathbb{w}; \mathsf{rand}), \\ \pi \leftarrow\!\!\$\ \mathsf{P}(\mathsf{crs}_{\mathsf{GS}}, (\mathbb{x}, \mathsf{com}), (\mathbb{w}, \mathsf{rand})) \end{array}\right\}$$
$$\{(\mathsf{com}', \pi') : (\mathsf{com}', \rho') \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Com}}(\mathsf{td}, \mathbb{x}), \pi \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{P}}(\mathsf{td}, \mathbb{x}, \rho)\}$$

Note that the proof system is zero-knowledge for *any multi-scalar multiplication and quadratic equations*, but for *pairing-product equations*, it is only zero-knowledge when $T_{\mathbb{G}_T} = \sum_k \mathsf{e}(T_{1,k}, T_{2,k})$ for some group elements $T_{1,k} \in \mathbb{G}_1, T_{2,k} \in \mathbb{G}_2$.

PAGH-PAGH FUNCTION FAMILIES. We particularly consider a specific instantiation of Pagh-Pagh [PP08, BHKN19] function families $\mathcal{F}$, mapping elements in $\mathcal{D} = \mathbb{Z}_p$ to $\mathcal{R} = \mathbb{Z}_p^3$. Each function $F_{\mathsf{key}} \in \mathcal{F}$ is defined by $\mathbb{Z}_p$-polynomials f_1, f_2, g_1, g_2, g_3 of degree $d = \Theta(\lambda)$ and two tables $T_1, T_2 \in (\mathbb{Z}_p^3)^S$. For $x \in \mathbb{Z}_p$, we compute $F_{\mathsf{key}}(x)$ as follows:

- Compute $y_1 \leftarrow f_1(x), y_2 \leftarrow f_2(x)$, and $z_i \leftarrow g_i(x)$ for $i \in [3]$.
- Truncate $\overline{y}_1 \leftarrow y_1 \pmod S, \overline{y}_2 \leftarrow y_2 \pmod S$.
- Return $T_1[\overline{y}_1] + T_2[\overline{y}_2] + (z_1, z_2, z_3)$. We will denote $T_{1,i}, T_{2,i}$ as the $\mathbb{Z}_p^S$ tables for each of the three positions $i = 1, 2, 3$.

The following lemma establishes the statistical PRF security of the function family $\mathcal{F}$ when we set $S = 8k$. We provide the proof in Section A following ideas in [BHKN19].

Lemma 4.4. *Let $d = d(\lambda) \geqslant 32, k = k(\lambda) \geqslant 2d, S = 8k$ be integers, and $p = p(\lambda)$ be a prime. Then, the function family $\mathcal{F}$ where containing functions mapping elements in $\mathcal{D} = \mathbb{Z}_p$ to $\mathcal{R} = \mathbb{Z}_p^3$ is such that any unbounded adversary $\mathcal{A}$ playing the PRF game of $\mathcal{F}$, making at most k queries, has*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{A}, \lambda) \leqslant \frac{k}{2^{d/2-6}} \ .$$

5 EARLT Construction from Bilinear Pairings

In this section, we give a construction of $\mathsf{EARLT} = \mathsf{EARLT}[\mathsf{GGen}]$ utilizing pairing-based group generated from GGen. Section 5.1 details the key generation and issuance protocols of the token dispensers and Section 5.2 describes the token showing algorithm along with the proof systems used. We discuss the efficiency of the scheme, possible extension, and comparison with existing lattice-based alternatives in Section 5.3.

Variable name	Description
λ	Security parameter
p	Prime-order group size
N	Number of tokens allowed per context string
ℓ_{cnt}	Bit size of $N = 2^{\ell_{\mathsf{cnt}}}$
ℓ_{ctxt}	Bit-length of context strings. Note: $2^{\ell_{\mathsf{ctxt}}+\ell_{\mathsf{cnt}}} < p$
k	Number of tokens allowed per one token dispenser
$d = \Theta(\lambda)$	Degree of polynomials f_1, f_2, g of key
$2^m = 8k$	Size of the tables T_1, T_2 which is a power of two.

Fig. 9. Variable names and descriptions.

5.1 Key Generation and Issuance

The key generation and issuance protocols are given in Figure 10, with parameters described in Figure 9. We now briefly summarize each component:

Setup: The setup algorithm generates (a) a CRS for $\mathsf{KZG}_{\mathsf{Ped}}$ commitment, (b) a CRS for vector commitment $\mathsf{VC}_{\mathsf{KZG}}$, (c) a CRS for Groth-Sahai proofs in *hiding mode*, (d) the CRS for the proof systems Π_{lin} and Π_{trunc} which will be used later on in showing protocol (see Section 5.2). Note that we write the setup of Π_{lin} and Π_{trunc} as taking $\mathsf{crs}_{\mathsf{GS}}$ as the language they are proving membership of depends on $\mathsf{crs}_{\mathsf{GS}}$.

Issuer's key generation: The issuer's key is a pair of secret and public key $(\mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}})$ of a structure-preserving signature scheme SPS.

User's key generation: The user's key is a pair $(\mathsf{sk}_{\mathsf{User}}, \mathsf{pk}_{\mathsf{User}})$ where $\mathsf{sk}_{\mathsf{User}} \in \mathbb{Z}_p$ is the discrete logarithm of $\mathsf{pk}_{\mathsf{User}} \in \mathbb{G}_1$.

Issuance protocol: The user first commits to the key of the function F_{key} which contains polynomials f_1, f_2, g_1, g_2, g_3 and tables $T_{1,i}, T_{2,i} \in \mathbb{Z}_p^{8k}$ for $i \in [3]$. The issuer then signs, using structure-preserving signature scheme SPS, these commitments along with a commitment C_γ of some rerandomization factor γ_0, γ_1, which will be used during the showing protocol to prevent collision in serial numbers. Finally, the user verifies the signature and output the token dispenser D, containing the key of the function F_{key} along with the commitments and the signature.

We separately use two CRS for $\mathsf{KZG}_{\mathsf{Ped}}$ and $\mathsf{VC}_{\mathsf{KZG}}$ to differentiate the usage of the two commitment schemes and modularize the security proof.

5.2 Showing Protocol

In this section, we give the showing, nonce-generation, verification and identification algorithms in Figures 11 and 12. The nonce algorithm simply samples a scalar r in $\mathbb{Z}_p^*$. At the high-level, the token showing for context ctxt proceeds by evaluating the Pagh-Pagh function on the key key and input $x = \mathsf{ctxt} \cdot 2^{\ell_{\mathsf{cnt}}} + \mathsf{cnt}$ where cnt is the number of the tokens shown for the context ctxt. Next, the showing algorithm will use Groth-Sahai commitment to commit to all the intermediate values $y_j, \overline{y}_j, t_{j,i}, z_i, \mathsf{cnt}$, the commitments C to key, and the signature σ. We call the commitment to all these values com, and denote the commitment for a specific variable xx as $\mathsf{com}_{\mathsf{xx}}$. We then prove that the evaluation is done correctly using multiple proof systems as we will describe below.

First, we define the relation $\mathcal{R}_{\mathsf{tok}}$ which constitutes statements that the proof systems in the protocol prove. This relation consists of the statements $\mathbb{x} = (\mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{VC}}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{pk}_{\mathsf{I}}, \mathsf{ctxt}, \mathsf{sn}, \mathsf{dbsp}, r, \mathsf{com})$ and witnesses $\mathbb{w} = (\widehat{\mathbb{w}}, \mathsf{rand})$ where

- $\mathsf{crs}_{\mathsf{KZG}} = (H, (X_{1,i}, \hat{X}_{1,i})_{i\in[d]}, X_{2,1})$ and $\mathsf{crs}_{\mathsf{VC}} = ((X'_{1,i})_{i\in[2^m]}, X'_{2,1})$
- $\widehat{\mathbb{w}} = (\mathsf{cnt}, \mathsf{sk}_{\mathsf{User}}, C, \sigma, C_\gamma, (y_j, \overline{y}_j, \mathsf{open}_{f,j})_{j\in[2]}, (t_{1,i}, t_{2,i}, z_i, \mathsf{open}_{T,1,i}, \mathsf{open}_{T,2,i}, \mathsf{open}_{g,i})_{i\in[3]})$
- $C = (C_{f,1}, C_{f,2}, (C_{g,i}, C_{T,1,i}, C_{T,2,i})_{i\in[3]})$
- $\mathsf{com} = \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \widehat{\mathbb{w}}; \mathsf{rand})$

Algorithm $\mathsf{Setup}(1^\lambda)$:

$\mathsf{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow\$ \mathsf{GGen}(1^\lambda)$
$\mathsf{crs}_{\mathsf{KZG}} \leftarrow\$ \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Setup}(\mathsf{par}, d)$
$\mathsf{crs}_{\mathsf{VC}} \leftarrow\$ \mathsf{VC}_{\mathsf{KZG}}.\mathsf{Setup}(\mathsf{par}, 2^m)$
$\mathsf{crs}_{\mathsf{GS}} \leftarrow\$ \mathsf{GS}.\mathsf{Setup}_{\mathsf{hide}}(\mathsf{par})$
$\mathsf{crs}_{\mathsf{lin}} \leftarrow\$ \Pi_{\mathsf{lin}}.\mathsf{Setup}(\mathsf{crs}_{\mathsf{GS}})$
$\mathsf{crs}_{\mathsf{trunc}} \leftarrow\$ \Pi_{\mathsf{trunc}}.\mathsf{Setup}(\mathsf{crs}_{\mathsf{GS}})$
$H_1, \ldots, H_4 \leftarrow\$ \mathbb{G}_1$
return $(\mathsf{par}, \mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{VC}}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{crs}_{\mathsf{lin}}, \mathsf{crs}_{\mathsf{trunc}}, (H_i)_{i\in[4]}, \mathsf{H}_{\mathsf{lin}}, \mathsf{H}_{\mathsf{trunc}})$

Algorithm $\mathsf{IKGen}(\mathsf{crs})$:

return $(\mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}) \leftarrow\$ \mathsf{SPS}.\mathsf{KeyGen}(\mathsf{par})$

Algorithm $\mathsf{UKGen}(\mathsf{crs})$:

return $(\mathsf{sk}_{\mathsf{User}} \leftarrow\$ \mathbb{Z}_p, \mathsf{pk}_{\mathsf{User}} \leftarrow \mathsf{sk}_{\mathsf{User}} G_1)$

Algorithm $\mathsf{I}(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{User}}, \mathsf{umsg} = C)$:

$\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1 \leftarrow\$ \mathbb{Z}_p^2$ // Rerandomization factor
// preventing collision in serial number
$C_\gamma \leftarrow \sum_{i=1}^{2} \gamma_{0,i} H_i + \gamma_{1,i} H_{2+i}$
$\sigma \leftarrow\$ \mathsf{SPS}.\mathsf{S}(\mathsf{sk}_{\mathsf{I}}, (\mathsf{pk}_{\mathsf{User}}, C, C_\gamma))$
return $\mathsf{imsg} = (\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, \sigma)$

Algorithm $\mathsf{User}_1(\mathsf{crs}, \mathsf{sk}_{\mathsf{User}}, \mathsf{pk}_{\mathsf{I}})$:

for $j \in [2]$ **do**
  $f_j \leftarrow\$ \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$
  $(C_{f,j}, \rho_{f,j}) \leftarrow\$ \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{KZG}}, f_j)$
for $i \in [3]$ **do**
  $g_i \leftarrow\$ \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$; $T_{1,i}, T_{2,i} \leftarrow\$ \mathbb{Z}_p^{8k}$
  $(C_{g,i}, \rho_{g,i}) \leftarrow\$ \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{KZG}}, g_i)$
  $(C_{T,1,i}, \mathsf{open}_{T,1,i}) \leftarrow\$ \mathsf{VC}_{\mathsf{KZG}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{VC}}, T_{1,i})$
  $(C_{T,2,i}, \mathsf{open}_{T,2,i}) \leftarrow\$ \mathsf{VC}_{\mathsf{KZG}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{VC}}, T_{2,i})$
$\mathsf{key} \leftarrow (f_1, f_2, (g_i, T_{1,i}, T_{2,i})_{i\in[3]})$
$C \leftarrow (C_{f,1}, C_{f,2}, (C_{g,i}, C_{T,1,i}, C_{T,2,i})_{i\in[3]})$
$\mathsf{open} \leftarrow (\mathsf{open}_{T,1,i}, \mathsf{open}_{T,2,i})_{i\in[3]}$
$\mathsf{st} \leftarrow (\mathsf{crs}, \mathsf{pk}_{\mathsf{I}}, \mathsf{sk}_{\mathsf{User}}, \mathsf{key}, C, \mathsf{open}, (\rho_{f,j})_{j\in[2]}, (\rho_{g,i})_{i\in[3]}))$
return (C, st)

Algorithm $\mathsf{User}_2(\mathsf{st}, \mathsf{imsg} = (\gamma_0, \gamma_1, \sigma))$

$C_\gamma \leftarrow \sum_{i=1}^{2} \gamma_{0,i} H_i + \gamma_{1,i} H_{2+i}$
if $\mathsf{SPS}.\mathsf{V}(\mathsf{pk}_{\mathsf{I}}, (\mathsf{pk}_{\mathsf{User}}, C, C_\gamma), \sigma) = 0$ **then abort**
$\mathsf{D} \leftarrow (\mathsf{pk}_{\mathsf{I}}, \mathsf{sk}_{\mathsf{User}}, \mathsf{key}, C, \mathsf{open}, (\rho_{f,i})_{i\in[2]}, (\rho_{g,i})_{i\in[3]}, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, C_\gamma, \sigma, \mathsf{ctr}[\cdot])$
// ctr keeps track of number of showings.
return D

Fig. 10. Setup, issuer's and user's key generation, and issuance protocol of $\mathsf{EARLT} = \mathsf{EARLT}[\mathsf{GGen}]$. The proof systems Π_{lin} and Π_{trunc} are given later in Section 5.2.

- The following equations are satisfied: Let $x = \mathsf{ctxt} \cdot N + \mathsf{cnt}$,

$$\mathsf{cnt} \in [0, N-1] \tag{R.1}$$
$$\mathsf{sn} = (t_{1,1} + t_{2,1} + z_1, t_{1,2} + t_{2,2} + z_2) + \mathsf{cnt} \cdot \boldsymbol{\gamma}_0 + \boldsymbol{\gamma}_1 \tag{R.2}$$
$$\mathsf{dbsp} = \mathsf{sk}_{\mathsf{User}} + r \cdot (t_{1,3} + t_{2,3} + z_3) \tag{R.3}$$
$$1 = \mathsf{SPS}.\mathsf{V}(\mathsf{pk}_{\mathsf{I}}, (\mathsf{sk}_{\mathsf{User}} G_1, C, C_\gamma), \sigma) \tag{R.4}$$
$$1 = \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{V}(\mathsf{crs}_{\mathsf{KZG}}, C_{f,j}, x, y_j, \mathsf{open}_{f,j}) \ , \forall j \in [2] \tag{R.5}$$
$$1 = \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{V}(\mathsf{crs}_{\mathsf{KZG}}, C_{g,i}, x, z_i, \mathsf{open}_{g,i}) \ , \forall i \in [3] \tag{R.6}$$
$$1 = \mathsf{VC}_{\mathsf{KZG}}.\mathsf{V}(\mathsf{crs}_{\mathsf{VC}}, C_{T,j,i}, \overline{y}_j, t_{j,i}, \mathsf{open}_{T,j,i}) \ , \forall j \in [2], i \in [3] \tag{R.7}$$
$$C_\gamma = \sum_{i=1}^{2} \gamma_{0,i} H_i + \gamma_{1,i} H_{2+i} \tag{R.8}$$
$$\overline{y}_j = y_j \pmod{8k} \ , \forall j \in [2] \tag{R.9}$$

Roughly, these equations say that the witness corresponds to the opening of the commitment com and the committed values correspond to the correct evaluation of F_{key} for the key $\mathsf{key} = (f_1, f_2, (T_{1,i}, T_{2,i}, g_i)_{i\in[3]})$ which is committed to and signed by sk_{I}. To prove each equation, we do the following:

- Equations (R.2) to (R.8) can be expressed as pairing-product equations (and equations of similar form), which can be proved using GS proofs.
- For Equation (R.1) and Equation (R.9), we use the proof system Π_{trunc} which we discuss in more detail later on. Equation (R.1) simply requires a range proof. For Equation (R.9), since we set $8k = 2^m$, this can be thought of as truncating the bits of y_1, y_2 when represented as integers in $[0, p-1]$.
- We additionally use a proof of knowledge Π_{lin} which proves knowledge of the openings of the commitments $\mathsf{com}_{y_j}, \mathsf{com}_{t_{j,i}}, \mathsf{com}_{\beta'_{f,j}}, \mathsf{com}_{\beta'_{g,i}}, \mathsf{com}_{z_i}$ and also $\mathsf{com}_{\boldsymbol{\gamma}_0}, \mathsf{com}_{\boldsymbol{\gamma}_1}$ as scalars. Note that these are the scalars which are evaluations and openings of the committed polynomials. In general, this is required for the security proof to reduce to binding of KZG commitment.

Algorithm $\mathsf{Show}(\mathsf{D}, \mathsf{ctxt}, r)$:

parse $(\mathsf{pk}_\mathsf{I}, \mathsf{sk}_\mathsf{User}, \mathsf{key}, C, \mathsf{open}, (\rho_{f,i})_{i\in[2]}, (\rho_{g,i})_{i\in[3]}, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, C_\gamma, \sigma, \mathsf{ctr}[\cdot]) \leftarrow \mathsf{D}$

parse $(C_{f,1}, C_{f,2}, (C_{g,i}, C_{T,1,i}, C_{T,2,i})_{i\in[3]}) \leftarrow C$

parse $(f_1, f_2, (g_i, T_{1,i}, T_{2,i})_{i\in[3]}) \leftarrow \mathsf{key}, (\mathsf{open}_{T,1,i}, \mathsf{open}_{T,2,i})_{i\in[3]} \leftarrow \mathsf{open}$

// **Initialize** $\mathsf{ctr}[\mathsf{ctxt}]$ **or abort if at least** N

if $\mathsf{ctr}[\mathsf{ctxt}] = \perp$ **then** $\mathsf{ctr}[\mathsf{ctxt}] \leftarrow 0$

if $\mathsf{ctr}[\mathsf{ctxt}] \geqslant N$ **then abort**

$\mathsf{cnt} \leftarrow \mathsf{ctr}[\mathsf{ctxt}]$; $x \leftarrow \mathsf{ctxt} \cdot N + \mathsf{cnt}$ // Setup counter and input

// **Compute intermediate values in the evaluation of** $\mathcal{F}_\mathsf{key}(x)$

for $j \in [2]$ **do**

$y_j \leftarrow f_j(x); \overline{y}_j \leftarrow y_j \pmod{8k}$

$\mathsf{open}_{f,j} = (\beta'_{f,j}, Q_{f,j}) \leftarrow \mathsf{KZG}_\mathsf{Ped}.\mathsf{Open}(\mathsf{crs}_\mathsf{KZG}, C_{f,j}, f_i, x, y_j, \rho_{f,j})$

for $i \in [3]$ **do**

$t_{1,i} \leftarrow T_{1,i}[\overline{y}_1]; t_{2,i} \leftarrow T_{2,i}[\overline{y}_2]; z_i \leftarrow g_i(x)$

$\mathsf{open}_{g,i} = (\beta'_{g,i}, Q_{g,i}) \leftarrow \mathsf{KZG}_\mathsf{Ped}.\mathsf{Open}(\mathsf{crs}_\mathsf{KZG}, C_{g,i}, g_i, x, z_i, \rho_{g,i})$

// **Compute serial number and double-spending equation**

$\mathsf{sn} \leftarrow (t_{1,1} + t_{2,1} + z_1, t_{1,2} + t_{2,2} + z_2) + \mathsf{cnt} \cdot \boldsymbol{\gamma}_0 + \boldsymbol{\gamma}_1$

$\mathsf{dbsp} \leftarrow \mathsf{sk}_\mathsf{User} + r \cdot (t_{1,3} + t_{2,3} + z_3)$

// **Commit to the witness**

$\hat{\mathbb{w}} \leftarrow (\mathsf{cnt}, \mathsf{sk}_\mathsf{User}, C, \sigma, C_\gamma, (y_j, \overline{y}_j, \mathsf{open}_{f,j})_{j\in[2]},$

$(t_{1,i}, t_{2,i}, z_i, \mathsf{open}_{T,1,i}[\overline{y}_1], \mathsf{open}_{T,2,i}[\overline{y}_2], \mathsf{open}_{g,i})_{i\in[3]}, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1)$

$(\mathsf{com}, \mathsf{rand}) \leftarrow\!\$\ \mathsf{GS}.\mathsf{Com}(\mathsf{crs}_\mathsf{GS}, \hat{\mathbb{w}})$; $\mathbb{w} \leftarrow (\hat{\mathbb{w}}, \mathsf{rand})$

// Denote $\mathsf{com}_{\mathsf{xx}}$ and $\mathsf{rand}_{\mathsf{xx}}$ as commitments to the value xx and corresponding randomness

// **Compute the proof of correct evaluation**

$\pi_\mathsf{GS} \leftarrow\!\$\ \mathsf{GS}.\mathsf{P}(\mathsf{crs}_\mathsf{GS}, (\text{Eq. R.2-R.8}, \mathsf{com}), \mathbb{w})$ // **Prove Eq. R.2-R.8**

$\mathbb{x}_\mathsf{lin} \leftarrow ((\mathsf{com}_{y_j}, \mathsf{com}_{\beta'_{f,j}})_{j,i\in[2]}, (\mathsf{com}_{t_{1,i}}, \mathsf{com}_{t_{2,i}}, \mathsf{com}_{z_i}, \mathsf{com}_{\beta'_{g,i}})_{i\in[3]}, \mathsf{com}_{\boldsymbol{\gamma}_0}, \mathsf{com}_{\boldsymbol{\gamma}_1})$

$\mathbb{w}_\mathsf{lin} \leftarrow ((y_j, \mathsf{rand}_{y_j}, \beta'_{f,j}, \mathsf{rand}_{\beta'_{f,j}})_{j\in[2]}, (z_i, \mathsf{rand}_{z_i}, \beta'_{g,i}, \mathsf{rand}_{\beta'_{g,i}}, (t_{j,i}, \mathsf{rand}_{t_{j,i}})_{j\in[2]})_{i\in[3]},$

$, \boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, \mathsf{rand}_{\boldsymbol{\gamma}_0}, \mathsf{rand}_{\boldsymbol{\gamma}_1})$

$\pi_\mathsf{lin} \leftarrow\!\$\ \Pi_\mathsf{lin}.\mathsf{P}^{\mathsf{H}_\mathsf{lin}}(\mathsf{crs}_\mathsf{lin}, \mathbb{x}_\mathsf{lin}, \mathbb{w}_\mathsf{lin})$ // **Prove knowledge of scalar openings**

$\mathbb{x}_\mathsf{trunc} \leftarrow (\mathsf{com}_\mathsf{cnt}, (\mathsf{com}_{y_j}, \mathsf{com}_{\overline{y}_j})_{j\in[2]}); \mathbb{w}_\mathsf{trunc} \leftarrow (\mathsf{cnt}, \mathsf{rand}_\mathsf{cnt}, (y_j, \mathsf{rand}_{y_j}, \overline{y}_j, \mathsf{rand}_{\overline{y}_j})_{j\in[2]})$

$\pi_\mathsf{trunc} \leftarrow\!\$\ \Pi_\mathsf{trunc}.\mathsf{P}^{\mathsf{H}_\mathsf{trunc}}(\mathsf{crs}_\mathsf{trunc}, \mathbb{x}_\mathsf{trunc}, \mathbb{w}_\mathsf{trunc})$ // **Prove Eq. R.1 and R.9**

$\mathsf{ctr}[\mathsf{ctxt}] \leftarrow \mathsf{ctr}[\mathsf{ctxt}] + 1$ // **Increment counter**

Update D to D' according to the new counter

return $((\mathsf{sn}, \tau = (\mathsf{dbsp}, \mathsf{com}, \pi = (\pi_\mathsf{GS}, \pi_\mathsf{lin}, \pi_\mathsf{trunc}))), \mathsf{D}')$

Algorithm $\mathsf{V}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt}, r, \tau = (\mathsf{sn}, \mathsf{com}, \pi))$:

parse $\mathbb{x}_\mathsf{lin}, \mathbb{x}_\mathsf{trunc}$ as in Show and **parse** $(\pi_\mathsf{GS}, \pi_\mathsf{lin}, \pi_\mathsf{trunc}) \leftarrow \pi$

return $\mathsf{GS}.\mathsf{V}(\mathsf{crs}_\mathsf{GS}, (\text{Eq. } R.2 - R.8, \mathsf{com}), \pi_\mathsf{GS}) \ \wedge$ // Verify Eq. R.2-R.8

$\Pi_\mathsf{lin}.\mathsf{P}(\mathsf{crs}_\mathsf{lin}, \mathbb{x}_\mathsf{lin}, \pi_\mathsf{lin}) \ \wedge \ \Pi_\mathsf{trunc}.\mathsf{P}(\mathsf{crs}_\mathsf{trunc}, \mathbb{x}_\mathsf{trunc}, \pi_\mathsf{trunc})$

Fig. 11. Showing and Verification algorithms.

Algorithm $\mathsf{nonce}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt})$:

return $r \leftarrow\!\$\ \mathbb{Z}_p^*$

Algorithm $\mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt}, r, r', \mathsf{sn}, \tau, \tau')$:

parse $(\mathsf{dbsp}, \mathsf{com}, \pi) \leftarrow \tau, (\mathsf{dbsp}', \mathsf{com}', \pi') \leftarrow \tau'$

return $\left(\mathsf{dbsp} - \dfrac{\mathsf{dbsp} - \mathsf{dbsp}'}{r - r'} r\right) G_1$

Fig. 12. Nonce generation and Identification algorithms.

GROTH-SAHAI PROOFS. We now describe how we use GS to prove equation (R.2) to equation (R.8). In particular, (R.2) and (R.3) are linear and quadratic equations over $\mathbb{Z}_p$, equations (R.4) to (R.7) are pairing-product equations, and equation (R.8) is a multi-scalar multiplication over $\mathbb{G}_1$. Ultimately, the elements in the witness will be committed as a commitment in $\mathbb{G}_1$ or $\mathbb{G}_2$ (or both). Accordingly, when $\mathsf{crs}_{\mathsf{GS}}$ is in binding mode, one can extract (through perfect F-knowledge property) the witness $\tilde{\mathbb{w}}$ containing

$$\begin{aligned}
&\widehat{\mathsf{cnt}}_1 \in \mathbb{G}_1, \widehat{\mathsf{cnt}}_2 \in \mathbb{G}_2, \mathsf{pk}_{\mathsf{User}} \in \mathbb{G}_1, C_\gamma \in \mathbb{G}_1, \boldsymbol{\Gamma}_0, \boldsymbol{\Gamma}_1 \in \mathbb{G}_2^2\\
&C = (C_{f,1}, C_{f,2}, (C_{g,i}, C_{T,1,i}, C_{T,2,i})_{i\in[3]}) \in \mathbb{G}_1^{11}, \sigma,\\
&(Y_j \in \mathbb{G}_1, \overline{Y}_{1,j} \in \mathbb{G}_1, \overline{Y}_{2,j} \in \mathbb{G}_2, \widehat{\mathsf{open}}_{f,j} = (B'_{f,j} \in \mathbb{G}_2, Q_{f,j} \in \mathbb{G}_1))_{j\in[2]},\\
&(Z_i \in \mathbb{G}_1, \widehat{\mathsf{open}}_{g,i} = (B'_{g,i} \in \mathbb{G}_2, Q_{g,i} \in \mathbb{G}_1), (\hat{T}_{j,i} \in \mathbb{G}_1, \mathsf{open}_{T,j,i} \in \mathbb{G}_1)_{j\in[2]})_{i\in[3]},
\end{aligned}$$

These group elements satisfy the following equations derived from (R.2) - (R.8).

$$\begin{aligned}
\mathsf{sn} \cdot G_T &= (T_{1,1} + T_{2,1} + Z_1, T_{1,2} + T_{2,2} + Z_2) + \mathsf{e}\left(\widehat{\mathsf{cnt}}_1, \boldsymbol{\Gamma}_0\right) + \mathsf{e}\left(G_1, \boldsymbol{\Gamma}_1\right)\\
\mathsf{dbsp} \cdot G_1 &= \mathsf{pk}_{\mathsf{User}} + r(T_{1,3} + T_{2,3} + Z_3)\\
1 &= \mathsf{SPS.V}(\mathsf{pk}_{\mathsf{I}}, (\mathsf{pk}_{\mathsf{User}}, C, C_\gamma), \sigma)\\
\mathsf{e}\left(\widehat{\mathsf{cnt}}_1, G_2\right) &= \mathsf{e}\left(G_1, \widehat{\mathsf{cnt}}_2\right)\\
\mathsf{e}\left(\overline{Y}_{1,j}, G_2\right) &= \mathsf{e}\left(G_1, \overline{Y}_{2,j}\right) && \forall j \in [2]\\
\mathsf{e}\left(C_{f,j} - Y_j, G_2\right) + \mathsf{e}\left(H, B'_{f,j}\right) &= \mathsf{e}\left(Q_{f,j}, X_{2,1} - \widehat{\mathsf{cnt}}_2 - \mathsf{ctxt} \cdot 2^{\ell_{\mathsf{cnt}}} G_2\right) && \forall j \in [2]\\
\mathsf{e}\left(C_{g,u} - Z_i, G_2\right) + \mathsf{e}\left(H, B'_{g,j}\right) &= \mathsf{e}\left(Q_{g,i}, X_{2,1} - \widehat{\mathsf{cnt}}_2 - \mathsf{ctxt} \cdot 2^{\ell_{\mathsf{cnt}}} G_2\right) && \forall i \in [3]\\
\mathsf{e}\left(C_{T,j,i} - \hat{T}_{j,i}, G_2\right) &= \mathsf{e}\left(\mathsf{open}_{T,j,i}, X'_{2,1} - \overline{Y}_{2,j}\right) && \forall j \in [2], i \in [3]\\
\mathsf{e}\left(C_\gamma, G_2\right) &= \textstyle\sum_{i=1}^{2} \mathsf{e}\left(H_i, \Gamma_{0,i}\right) + \mathsf{e}\left(H_{2+i}, \Gamma_{1,i}\right) && (1)
\end{aligned}$$

We denote (1) as all of the equations above. For convenience, we say $(\mathbb{x}, \tilde{\mathbb{w}}) \in \widetilde{\mathcal{R}}_{\mathsf{tok}}$ for statements $\mathbb{x}$ of the same format as $\mathcal{R}_{\mathsf{tok}}$ and $\tilde{\mathbb{w}}$ satisfying (1).

PROOF Π_{lin}. This proof, given in Figure 13, is a simple proof of knowledge for linear homomorphism over $\mathbb{G}_1$ and $\mathbb{G}_2$, i.e., for the following relation:

$$\begin{aligned}
\mathcal{R}_{\mathsf{lin},\mathsf{crs}_{\mathsf{GS}}} := \{&(\mathbb{x} = (\mathsf{com}_{i,1} \in \mathbb{G}_1^2)_{i\in[l_1]}, (\mathsf{com}_{i,2} \in \mathbb{G}_2^2)_{i\in[l_2]}),\\
&\mathbb{w} = ((y_{i,1} \in \mathbb{Z}_p, \mathsf{rand}_{i,1} \in \mathbb{Z}_p^2)_{i\in[l_1]}, (y_{i,2} \in \mathbb{Z}_p, \mathsf{rand}_{i,2} \in \mathbb{Z}_p^2)_{i\in[l_2]})) :\\
&\forall i \in [l_1] : \mathsf{com}_{i,1} = \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, y_{i,1}; \mathsf{rand}_{i,1}) \wedge\\
&\forall i \in [l_2] : \mathsf{com}_{i,2} = \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, y_{i,2}; \mathsf{rand}_{i,2})\} .
\end{aligned}$$

For the token showing algorithm, we specifically use Π_{lin} to show knowledge of openings to the commitments $((\mathsf{com}_{y_j}, \mathsf{com}_{\beta'_{f,j}})_{j,\in[2]}, (\mathsf{com}_{t_{1,i}}, \mathsf{com}_{t_{2,i}}, \mathsf{com}_{z_i}, \mathsf{com}_{\beta'_{g,i}})_{i\in[3]}, \mathsf{com}_{\boldsymbol{\gamma}_0}, \mathsf{com}_{\boldsymbol{\gamma}_1})$, which corresponds to the $\mathbb{Z}_p$ scalars in the openings of the commitment $C_{g,i}, C_{f,j}, C_{T,j,i}$, and C_γ.

We note that Π_{lin} is correct, perfect zero-knowledge, and multi-proof rewinding extractable as stated in the following lemma. Note that we only need the lemma when $\mathsf{crs}_{\mathsf{GS}}$ is generated in binding mode.

Lemma 5.1. *Let* GGen *be a group generator outputting groups of prime-order* $p = p(\lambda)$*. The proof system* Π_{lin} *is correct and perfect zero-knowledge , i.e., there exists a simulator* Sim *such that for any adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{zk}_{\Pi_{\mathsf{lin}},\mathsf{Sim}}(\mathcal{A}, \lambda) = 0$*. Also, let* $\mathcal{D}_\lambda$ *be a distribution of inputs that contains* $\mathsf{par} \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda)$ *and* $\mathsf{crs}_{\mathsf{lin}} = (\mathsf{crs}_{\mathsf{GS}}, Z \in \mathbb{G}_1)$ *with* $\mathsf{crs}_{\mathsf{GS}}$ *generated from* $\mathsf{GS.Setup}_{\mathsf{bind}}$ *and* $Z \leftarrow\!\!\$\ \mathbb{G}_1$ *and some auxiliary input* inp'*. Let* $\mathcal{A}$ *be an algorithm taking an input* $\mathsf{inp} \leftarrow\!\!\$\ \mathcal{D}$ *and making at most* $Q_{\mathsf{H}} = Q_{\mathsf{H}}(\lambda)$ *queries to* $\mathsf{H}_{\mathsf{lin}}$ *modeled as a random oracle. Denote* $\mathsf{acc}(\mathcal{A}, \lambda)$ *as the probability (over* $\mathsf{inp} \leftarrow\!\!\$\ \mathcal{D}$*, the random oracle* H*'s outputs, and the random coins* $\rho_{\mathcal{A}}$ *sampled from* $\mathcal{R}_{\mathcal{A}}$ *of* $\mathcal{A}$*) such that* $\mathcal{A}$ *outputs* $((\mathbb{x}_i, \pi_i)_{i\in[L]}, \mathsf{aux})$ *where all of the pairs* $(\mathbb{x}_i, \pi_i)$ *verify.*

Algorithm $\Pi_{\mathsf{lin}}.\mathsf{P}(\mathsf{crs}_{\mathsf{lin}} = (\mathsf{crs}_{\mathsf{GS}}, Z), \mathbb{x}, \mathbb{w})$

parse $((\mathsf{com}_{i,1})_{i\in[l_1]}, (\mathsf{com}_{i,2})_{i\in[l_2]}) \leftarrow \mathbb{x}$
parse $((y_{i,1} \in \mathbb{Z}_p, \mathsf{rand}_{i,1})_{i\in[l_1]},$
$\quad (y_{i,2} \in \mathbb{Z}_p, \mathsf{rand}_{i,2})_{i\in[l_2]}) \leftarrow \mathbb{w}$
$r_{1,1}, \ldots, r_{l_1,1}, r_{1,2}, \ldots, r_{l_2,2}, c_1, s \leftarrow\$ \mathbb{Z}_p$
$\rho_{1,1}, \ldots, \rho_{l_1,1}, \rho_{1,2}, \ldots, \rho_{l_2,2} \leftarrow\$ \mathbb{Z}_p^2$
for $j \in [2], i \in [l_j]$ **do**
$\quad A_{i,j} \leftarrow \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, r_{i,j}; \rho_{i,j})$
$R \leftarrow sG_1 - c_1 Z$
$c \leftarrow \mathsf{H}(\mathbb{x}, (A_{i,j})_{j\in[2],i\in[l_j]}, R)$; $c_0 \leftarrow c/c_1$
for $j \in [2], i \in [l_j]$ **do**
$\quad z_{i,j} \leftarrow r_{i,j} + c_0 \cdot y_{i,j}; \zeta_{i,j} \leftarrow \rho_{i,j} + c_0 \cdot \mathsf{rand}_{i,j}$
return $(c_0, c_1, (z_{i,j}, \zeta_{i,j})_{j\in[2],i\in[l_j]}, s)$

Algorithm $\Pi_{\mathsf{lin}}.\mathsf{Setup}(\mathsf{crs}_{\mathsf{GS}})$:

$Z \leftarrow\$ \mathbb{G}_1$
return $\mathsf{crs}_{\mathsf{lin}} = (\mathsf{crs}_{\mathsf{GS}}, Z)$

Algorithm $\Pi_{\mathsf{lin}}.\mathsf{V}(\mathsf{crs}_{\mathsf{lin}}, \mathbb{x}, \pi)$

parse $((\mathsf{com}_{i,1})_{i\in[l_1]}, (\mathsf{com}_{i,2})_{i\in[l_2]}) \leftarrow \mathbb{x}$
parse $(c_0, c_1, (z_{i,j}, \zeta_{i,j})_{j\in[2],i\in[l_j]}, s) \leftarrow \pi$
for $j \in [2], i \in [l_j]$ **do**
$\quad A_{i,j} \leftarrow \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, r_{i,j}; \rho_{i,j}) - c_0 \cdot \mathsf{com}_{i,j}$
$R \leftarrow sG_1 - c_1 Z$
return $c_0 + c_1 = \mathsf{H}(\mathbb{x}, (A_{i,j})_{j\in[2],i\in[l_j]}, R)$

Fig. 13. Proof system Π_{lin}. Note that the setup takes $\mathsf{crs}_{\mathsf{GS}}$ instead of 1^λ since the language depends on Groth-Sahai CRS. One can view the actual setup algorithm as sampling the group $\mathsf{par} \leftarrow\$ \mathsf{GGen}(1^\lambda)$ and running $\mathsf{GS.Setup}$ and $\Pi_{\mathsf{lin}}.\mathsf{Setup}$.

Then, there exists an efficient extractor $\mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}}$ *taking as input* $\mathsf{inp} \leftarrow\$ \mathcal{D}$ *and outputs* $(\mathbb{w}_i)_{i\in[L]}$*, such that there exists an adversary* $\mathcal{B}_{\mathrm{dlog}}$ *such that*

$$\mathsf{Pr}\left[\begin{array}{l|l} \forall i \in [L]: & \mathsf{inp} = (\mathsf{par}, \mathsf{crs}_{\mathsf{lin}} = (\mathsf{crs}_{\mathsf{GS}}, Z), \mathsf{inp}') \leftarrow\$ \mathcal{D} \\ ((\mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}_{\mathsf{lin},\mathsf{crs}_{\mathsf{GS}}} \wedge & \rho_{\mathcal{A}} \leftarrow\$ \mathcal{R}_{\mathcal{A}} \\ \Pi_{\mathsf{lin}}.\mathsf{V}^{\mathsf{H}_{\mathsf{lin}}}(\mathsf{crs}_{\mathsf{lin}}, \mathbb{x}_i, \pi_i) = 1) & \mathsf{out} = ((\mathbb{x}_i, \pi_i)_{i\in[L]}, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{H}_{\mathsf{lin}}}(\mathsf{inp}; \rho_{\mathcal{A}}) \\ & (\mathbb{w}_i)_{i\in[L]} \leftarrow\$ \mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}}(\mathsf{inp}, \mathsf{out}, \boldsymbol{h}; \rho_{\mathcal{A}}) \end{array}\right]$$
$$\geqslant \frac{\mathsf{acc}(\mathcal{A}, \lambda)}{8} - \mathsf{Adv}_{\mathsf{GGen}}^{\mathrm{dlog}}(\mathcal{B}_{\mathrm{dlog}}, \lambda)\ ,$$

where $\boldsymbol{h}$ *denotes the random oracle outputs during the run of* $\mathcal{A}$*. Additionally,* Ext *and* $\mathcal{B}_{\mathrm{dlog}}$ *run in time at most* $\frac{8L^2 Q_{\mathsf{H}}}{\mathsf{acc}} \ln(\frac{8L}{\mathsf{acc}})$ *times that of* $\mathcal{A}$*.*

Proof (of Lemma 5.1). Correctness follows by inspection, and perfect zero-knowledge follows by setting up Z with a trapdoor $z \in \mathbb{Z}_p$ (the CRS is still identically distributed) and using z to compute the proof for statement $((\mathsf{com}_{i,1})_{i\in[l_1]}, (\mathsf{com}_{i,2})_{i\in[l_2]})$ by sampling $c_0 \leftarrow\$ \mathbb{Z}_p$ and compute $A_{i,j} \leftarrow \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, z_{i,j}; \zeta_{i,j}) - c_0 \mathsf{com}_{i,j}$ with $z_{i,j} \leftarrow\$ \mathbb{Z}_p, \zeta_{i,j} \leftarrow\$ \mathbb{Z}_p^2$ and compute $R \leftarrow rG_1$ with $s = r + (c - c_0) \cdot z$. Given c, the distribution of $(c_0, c_1, (A_{i,j}, z_{i,j}, \zeta_{i,j}), R,)$ is identical to that of the actual prover. Hence, we have perfect zero-knowledge.

The property of the extractor follows from the multi-forking lemma Lemma 2.2 as we will show now. Without loss of generality, we assume that $\mathcal{A}$ already makes the query to be made at verification. Now, we define a wrapper $\mathcal{A}'$ which takes as input $\mathsf{inp} \leftarrow\$ \mathcal{D}$ (and a sampled random coin $\rho_{\mathcal{A}}$) and a list $h_1, \ldots, h_{Q_{\mathsf{H}}} \in \mathbb{Z}_p$ and runs $\mathcal{A}$ on input inp and program the random oracle $\mathsf{H}_{\mathsf{lin}}$ using $h_1, \ldots, h_{Q_{\mathsf{H}}}$ (by incrementing a counter ctr and using h_{ctr} on a new query). On the output $((\mathbb{x}_i, \pi_i)_{i\in[L]}, \mathsf{aux})$ of $\mathcal{A}$, $\mathcal{A}'$ checks if all proofs are valid, and if not abort, set $F \leftarrow \varnothing$, otherwise, set F as the set of indices of RO queries corresponding to each of the L proofs $\mathcal{A}$ outputs. Then, it returns $(F, (\mathbb{x}_i, \pi_i)_{i\in[L]}, \mathsf{aux})$. Then, define the extractor $\mathsf{Ext}_{\mathsf{lin}}$ as follows:

- It takes as input $\mathsf{inp} = (\mathsf{par}, \mathsf{crs}_{\mathsf{lin}} = (\mathsf{crs}_{\mathsf{GS}}, Z), \mathsf{inp}')$, and $((\mathbb{x}_i, \pi_i)_{i\in[L]}, \mathsf{aux}), \boldsymbol{h}; \rho_{\mathcal{A}}$.
- Run $\mathcal{A}'$ on input $(\mathsf{inp}, \boldsymbol{h}; \rho_{\mathcal{A}})$ and get the output $(F, \mathsf{out}, \mathsf{aux})$ which should be the same as $((\mathbb{x}_i, \pi_i)_{i\in[L]}, \mathsf{aux})$, since $\mathcal{A}'$ essentially runs $\mathcal{A}$ again with the same randomness and RO outputs.
- If $F = \perp$, return $\perp$.
- Rewind $\mathcal{A}'$ as $\mathsf{MFork}^{\mathcal{A}'}(\mathsf{inp})$ would if the first run of $\mathcal{A}'$ uses $\boldsymbol{h}$. (Refer to the description of MFork in Lemma 2.2 after the second bulletpoint.) At the end of this process, the extractor will obtain $\mathsf{out}' = (h_i', \mathbb{x}_i', \pi_i')_{i\in[L]}$.

- For each $i \in [L]$, parse
  - $((\mathsf{com}_{i,j,1})_{j\in[l_1]}, (\mathsf{com}_{i,j,2})_{j\in[l_2]}) \leftarrow \mathbb{x}_i$
  - $(c_{i,0}, c_{i,1}, (z_{i,k,j}, \zeta_{i,k,j})_{j\in[2],k\in[l_j]}, s_i) \leftarrow \pi_i$
  - $((\mathsf{com}'_{i,j,1})_{j\in[l_1]}, (\mathsf{com}'_{i,j,2})_{j\in[l_2]}) \leftarrow \mathbb{x}'_i$. Note that by how MFork and $\mathcal{A}'$ are defined $\mathbb{x}_i = \mathbb{x}'_i$ as they corresponds to the same RO query from $\mathcal{A}$.
  - $(c'_{i,0}, c'_{i,1}, (z'_{i,k,j}, \zeta'_{i,k,j})_{j\in[2],k\in[l_j]}, s'_i) \leftarrow \pi'_i$
- If $c_{i,0} = c'_{i,0}$ for some $i \in [L]$, return $\perp$.
- Otherwise, set $\mathbb{w}_i = (c_{i,0} - c'_{i,0})^{-1} \cdot (z_{i,k,j} - z'_{i,k,j}, \zeta_{i,k,j} - \zeta'_{i,k,j})_{j\in[2],k\in[l_j]}$ and return $(\mathbb{w}_i)_{i\in[L]}$.

Also, note that when $F \neq \perp$, we have that all the proofs verify and $h_i \neq h'_i$ for all $i \in [L]$. Now, we consider the event that for some $i \in [L]$, $c_{i,0} = c'_{i,0}$ and denote this event as Bad. Since $h_i \neq h'_i$, we have that $c_{i,1} \neq c'_{i,1}$, which allows us to recover the discrete log $z = (s_i - s'_i)/(c_{i,1} - c'_{i,1})$ as the proofs π_i, π'_i correspond to the same random oracle query (hence the same $R_i = s_i G_1 - c_{i,1} Z = s'_i G_1 - c'_{i,1} Z$). We can see that there exists $\mathcal{B}$ playing DL game with running time as in the lemma statement such that $\mathsf{Pr}[\mathsf{Bad}] \leqslant \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{GGen}}(\mathcal{B}, \lambda)$.

If Bad does not occur, we can easily see that $\mathbb{w}_i$ is a witness for $\mathbb{x}_i$ as the commitment scheme is linearly homomorphic. The bound then follows from Lemma 2.2. □

PROOF FOR CORRECT TRUNCATION AND MORE Π_{trunc}. We give a proof system for the relation $\mathcal{R}_{\mathsf{trunc}}$ defined as

$$\mathcal{R}_{\mathsf{trunc,crs}} := \left\{ \begin{array}{l} (\mathbb{x} = (\mathsf{com}_{\mathsf{cnt}}, (\mathsf{com}_{y_j}, \mathsf{com}_{\overline{y}_j})_{j\in[2]}), \\ \mathbb{w} = (\mathsf{cnt}, \mathsf{rand}_{\mathsf{cnt}}, \\ \quad (y_j, \mathsf{rand}_{y_j}, \overline{y}_j, \mathsf{rand}_{\overline{y}_j})_{j\in[2]}) \end{array} : \begin{array}{l} \mathsf{com}_{\mathsf{cnt}} = (\mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \mathsf{cnt}; \mathsf{rand}_{\mathsf{cnt}}) \wedge \\ \mathsf{cnt} \in [0, N-1] \wedge \\ (\forall j \in [2] : \mathsf{com}_{y_j} = \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, y_j; \mathsf{rand}_{y_j}) \wedge \\ \quad \mathsf{com}_{\overline{y}_j} = \mathsf{GS.Com}(\mathsf{crs}_{\mathsf{GS}}, \overline{y}_j; \mathsf{rand}_{\overline{y}_j}) \wedge \\ \quad y_j \in [0, p-1] \ \wedge \ \overline{y}_j \in [0, 2^m - 1] \\ \quad \wedge \ \overline{y} \equiv y \pmod{2^m}) \end{array} \right\}.$$

We included the commitment to the counter cnt as well to make the token size more compact instead of proving them separately. As alluded to earlier, this relation shows that cnt is in the range $[0, N-1]$ and $\overline{y}$ is exactly the integer $y \in [0, p-1]$ truncated to only m bits. (Just for the high-level description, we forgo the subscript $j \in [2]$) First, we write $y = \sum_{i=0}^{n} b_i 2^i \in [0, p-1]$ with $b_i \in \{0,1\}$ for all $i \in [0, n]$. To prove that $\overline{y} = y \pmod{2^m}$ using group of prime-order p, one would do the following at a high-level:

- Commit to the bits $b_0, \ldots, b_n$.
- Prove that $\overline{y} = \sum_{i=0}^{m-1} b_i 2^i \pmod{p}$ and $y = \sum_{i=0}^{n} b_i 2^i \pmod{p}$.
- Additionally, we need to show that $\sum_{i=0}^{n} b_i 2^i \in [0, p-1]$. Note that this is *very crucial* as the prime-order group structure does *not* ensure that the committed bits sum up to an integer in the range $[0, p-1]$ due to overflowing in modulo p.[6] In this case, one needs to prove the following: (a) $b_n = 0$ or (b) $\sum_{i=0}^{n-1} b_i 2^i \in [0, p-1-2^n]$. For (b), we additionally commits to bits $b'_0, \ldots, b'_{n'-1}$ and prove that if $b_n = 1$, then $p - 1 - y = \sum_{i=0}^{n'-1} b'_i 2^i \pmod{p}$ where $n' = \lceil \log(p - 2^n) \rceil$.

The following lemma shows that such condition is the necessary and sufficient to ensure that $\sum_{i=0}^{n} b_i 2^i \in [0, p-1]$.

Lemma 5.2. *Let p be an odd prime, $n = \lfloor \log p \rfloor$ and $n' = \lceil \log(p - 2^n) \rceil$. Also, let $b_0, \ldots, b_n \in \{0,1\}$ and $y = \sum_{i=0}^{n} b_i 2^i \pmod{p}$. Then, $\sum_{i=0}^{n} b_i 2^i \in [0, p-1]$ if and only if $(b_n = 0) \ \vee \ (b_n = 1 \ \wedge \ (p - 1 - y) \pmod{p} \in [0, 2^{n'} - 1])$.*

Proof. ($\Rightarrow$) Suppose $\sum_{i=0}^{n} b_i 2^i \in [0, p-1]$, then this is straightforward as either $b_n = 0$ or if $b_n = 1$, $\sum_{i=0}^{n-1} b_i 2^i = y - 2^n \in [0, p-1-2^n]$. Since $p - 2^n \leqslant 2^{n'}$ by definition of n', $p - 1 - y = p - 1 - 2^n - (y - 2^n) \leqslant 2^{n'} - 1$.

($\Leftarrow$) Suppose $b_n = 0$ or $(b_n = 1 \ \wedge \ (p-1-y) \pmod{p} \in [0, 2^{n'} - 1])$. For the first case, it is already implied that $\sum_{i=0}^{n} b_i 2^i \in [0, p-1]$. For the latter, suppose for contradiction that $\sum_{i=0}^{n} b_i 2^i \geqslant p$. Then, $y = \sum_{i=0}^{n} b_i 2^i$

[6] For example, both the bit-decomposition of 0 and p sum up to the same $\mathbb{Z}_p$ element.

$(\text{mod } p) = \sum_{i=0}^{n} b_i 2^i - p$ (the last equality is saying that the representation of y is exactly that of the integer $\sum_{i=0}^{n} b_i 2^i - p$). Additionally, we have that $y < 2^{n+1} - p$.

Now, consider $(p - 1 - y) \pmod p > p - 1 - 2^{n+1} + p = 2(p - 2^n) - 1 \geqslant 2^{n'-1+1} - 1 = 2^{n'} - 1$. The first inequality follows from $y < 2^{n+1} - p$. The second inequality follows from the definition of n'. Hence, proving the statement. □

Hence, the constraints defined in the relation $\mathcal{R}_{\mathsf{trunc},\mathsf{crs}}$ boils down to proving the following equations over the vector $\boldsymbol{v} := (\mathsf{cnt}, y_1, y_2, \overline{y}_1, \overline{y}_2, \boldsymbol{b}_{\mathsf{cnt}}, \boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}'_0, \boldsymbol{b}'_1)$ where $\boldsymbol{b}_{\mathsf{cnt}} \in \mathbb{Z}_p^{\ell_{\mathsf{cnt}}}, \boldsymbol{b}_0, \boldsymbol{b}_1 \in \mathbb{Z}_p^{n+1}, \boldsymbol{b}'_0, \boldsymbol{b}'_1 \in \mathbb{Z}_p^{n'}$. Let $\hat{\boldsymbol{b}} = (\boldsymbol{b}_{\mathsf{cnt}}, \boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}'_0, \boldsymbol{b}'_1)$

$$\mathbf{0} = \hat{\boldsymbol{b}} \circ (\mathbf{1} - \hat{\boldsymbol{b}}) \tag{2}$$

$$0 = \sum_{i=0}^{\ell_{\mathsf{cnt}}-1} b_{\mathsf{cnt},i} 2^i - \mathsf{cnt} \tag{3}$$

$$0 = \sum_{i=0}^{n} b_{j,i} 2^i - y_j \qquad \forall j \in [2] \tag{4}$$

$$0 = \left(\sum_{i=0}^{n'-1} b'_{j,i} 2^i + 1 + y_j\right) b_{j,n} \qquad \forall j \in [2] \tag{5}$$

$$0 = \sum_{i=0}^{m-1} b_{j,i} 2^i - \overline{y}_j \qquad \forall j \in [2] \tag{6}$$

The first equation shows that $\boldsymbol{b}_{\mathsf{cnt}}, \boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}'_0, \boldsymbol{b}'_1$ are vector of bits ($\circ$ denotes element-wise multiplication). The second equation shows that $\mathsf{cnt} \in [0, N-1]$. The third equation, combined with Lemma 5.2, shows that $y_j = \sum_{i=0}^{n} b_{j,i} 2^i \in [0, p-1]$. The fourth equation shows that $\overline{y}_j = y \pmod{2^m}$.

The above consists of $N_{\mathsf{trunc}} = 2(n+n'+1)+\ell_{\mathsf{cnt}}+2$ quadratic equations over $\boldsymbol{v}$ and 5 more linear equations. These can be proved using efficient proofs for arithmetic circuits, e.g., Bulletproofs [BBB+18] or Compressed Σ-protocol [AC20]. In our setting, we have two additional inconveniences to resolve: (1) our commitments are Groth-Sahai commitments instead of Pedersen, and (2) we require statistical zero-knowledge (and this is *without* assuming limited query to the random oracle model). Fortunately, the framework of [AC20] allow us to easily integrate these aspects into the proof system.

To this end, we construct the underlying interactive proof $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$ (given in Figure 14), make it zero-knowledge (instead of HVZK), and derive Π_{trunc} by applying the Fiat-Shamir transformation [FS87]. For $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$, we give a high-level idea from the framework of [AC20] as follows:

1. Compute a Pedersen commitment P of the vector $\boldsymbol{v}$ while proving that $\mathsf{cnt}, y_1, y_2, \overline{y}_1, \overline{y}_2$ which is committed to in com are contained in $\boldsymbol{v}$.
2. Apply the proof for arithmetic circuit on $\boldsymbol{v}$. In particular, the prover samples random $\mathbb{Z}_p$-polynomials f, g of degree at most N_{trunc} such that $f(i) = \hat{\boldsymbol{b}}_i = 1 - g(i)$ for $i \in [2(n+n'+1)+\ell_{\mathsf{cnt}}]$ and $f(N_{\mathsf{trunc}} - j + 1) = \sum_{i=0}^{n'-1} b'_{j,i} 2^i + 1 + y, g(N_{\mathsf{trunc}} - j + 1) = b_{j,n}$ for $j \in [2]$ (note that $f(0), g(0)$ are uniformly random). The prover additionally commits to $f(0), g(0)$ and the evaluation of a polynomial $h(x) = f(x) \cdot g(x)$ of degree at most $2N_{\mathsf{trunc}}$ on $x = 0, \ldots, 2N_{\mathsf{trunc}}$. The verifier then sends a random challenge $c \in \mathbb{Z}_p$ for which the prover replies with $z_1 = f(c), z_2 = g(c), z_3 = h(c)$.
   This linearizes the multiplications as the prover now proves only linear equations over the $\boldsymbol{v}$ and $f(0), g(0), h(0), h(1), \ldots, h(2N_{\mathsf{trunc}})$. In particular, the prover now only proves that the committed vector can be linearly combined into z_1, z_2, z_3 with respect to the correct interpolation equality, and for each output of the circuit (in our case, each multiplication gate), the affine linear combination over the inputs $\boldsymbol{v}$ and outputs of the multiplication gates $h(i)$ (for $i \in [2N_{\mathsf{trunc}} + 7]$) satisfy the claimed output. In our case, this means the outputs of the multiplication are zero, and that z_1, z_2, z_3 are actually evaluation of the committed f, g, h. Note that we can optimize by not committing to $h(1), \ldots, h(N_{\mathsf{trunc}})$ since we claim they are 0. (See [AC20] for more detail.)
   To prove these linear equations, we use an interactive proof system Π_{Nullity} (as defined in [AC20]) proving knowledge of a witness $\boldsymbol{v}'$ for the following relation where $\{L_i(\boldsymbol{x})\}$ is a set of affine functions. Formally, it proves knowledge of a witness in the following relation

$$\mathcal{R}_{\mathsf{Nullity}} := \{((\boldsymbol{H}_{\mathsf{trunc}}, P, \{L_i(\boldsymbol{x})\}), \boldsymbol{v}) : P = \sum v_i H_{\mathsf{trunc},i} \ \wedge \ \forall i : L_i(\boldsymbol{v}) = 0\} .$$

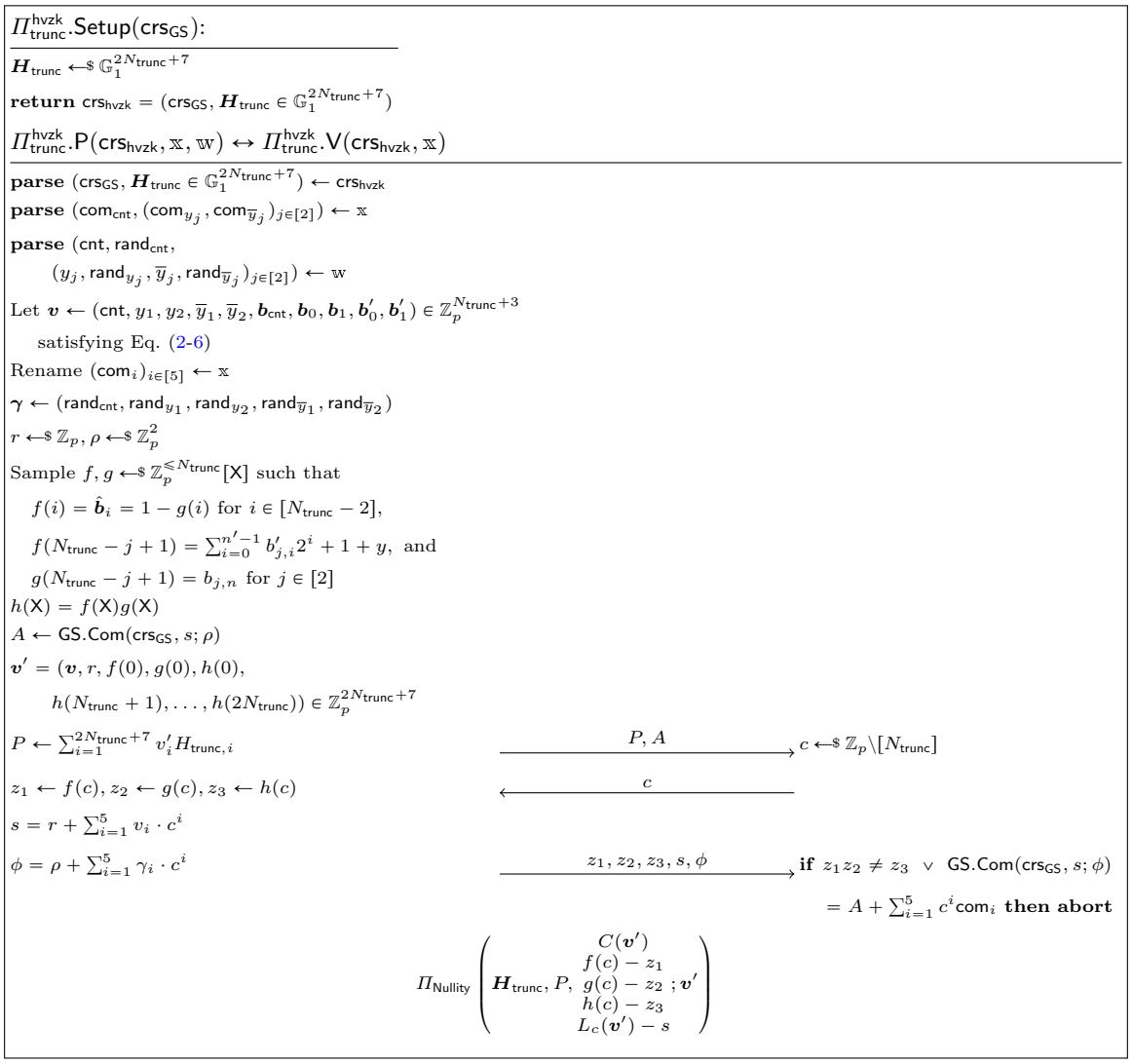

Fig. 14. Interactive proof $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$. Note: $N_{\mathsf{trunc}} = 2(n + n' + 1) + \ell_{\mathsf{cnt}} + 2$. Note that f, g are sampled randomly such that $f(i), g(i)$ corresponds to the left and right input of multiplication gates in C, which is defined by Eq. (2-6). Also, $L_c(\boldsymbol{v}') := s + \sum_{i=1}^{5} c^i v'_i$. We define the setup takes $\mathsf{crs}_{\mathsf{GS}}$ instead of 1^λ since the language depends on the Groth-Sahai CRS.

To make the proof system zero-knowledge instead of HVZK, we include in the CRS a group element $Z \in \mathbb{G}_1$ and let the prover either prove knowledge of the discrete log z or that the statement is true. Denote this proof system as $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ (given in Figure 15). We note that the first verifier challenge for $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$ is not uniformly random in $\mathbb{Z}_p$ but $\mathbb{Z}_p \backslash \{1, \ldots, N_{\mathsf{trunc}}\}$. We can think of this first challenge as being uniform in $[0, p - N_{\mathsf{trunc}} - 1]$ as well as $c_1, c_{1,1} \in [0, p - N_{\mathsf{trunc}} - 1]$ in the description of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$. Then, compute $c_{1,0} = c_1 - c_{1,1}$ in modulo $p - N_{\mathsf{trunc}}$ and map it back into $\mathbb{Z}_p \backslash \{1, \ldots, N_{\mathsf{trunc}}\}$ (there exists such a bijection by simple enumeration of elements), so that $c_{1,0}$ is still uniformly random in the same domain $[0, p - N_{\mathsf{trunc}} - 1]$. The ZK simulator for $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ will generate Z with the discrete log z and simulate the protocol using z. Finally, we denote the non-interactive proof Π_{trunc} as the Fiat-Shamir transformed [FS87] version of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$.

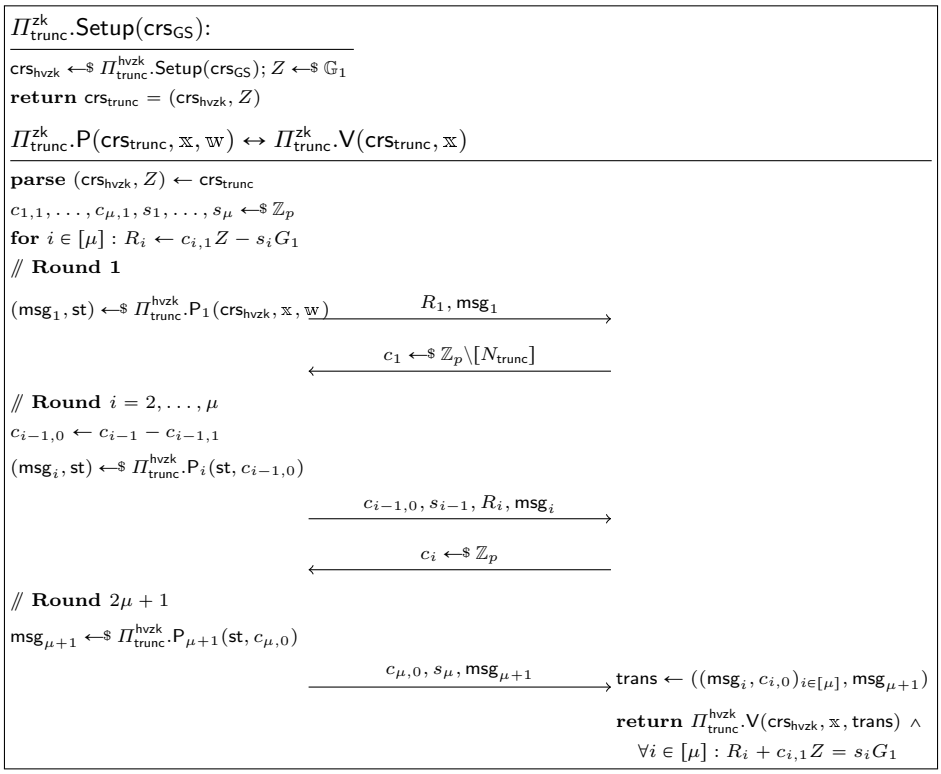

Fig. 15. Interactive zero-knowledge proof $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ constructed using $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$. Similar to $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$, we define the setup takes $\mathsf{crs}_{\mathsf{GS}}$ instead of 1^λ since the language depends on the Groth-Sahai CRS. For the first challenges $c_1, c_{1,0}, c_{1,1}$, we view it as being in $[0, p - N_{\mathsf{trunc}} - 1]$ (via a simple bijection between $[0, p - N_{\mathsf{trunc}} - 1]$ and $\mathbb{Z}_p \backslash [N_{\mathsf{trunc}}]$). Then, when we compute $c_{1,0} = c_1 - c_{1,1}$ we do so in modulo $p - N_{\mathsf{trunc}}$ and map it back into $\mathbb{Z}_p \backslash [N_{\mathsf{trunc}}]$.

Now, we define a relaxed relation

$$\widetilde{\mathcal{R}}_{\mathsf{trunc},\mathsf{crs}} := \{(\mathbb{x}, \mathbb{w}) : (\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{trunc}} \ \vee \ (\mathsf{crs}, \mathbb{w}) \in \mathcal{R}_{\mathrm{dlog}}\} ,$$

where $\mathcal{R}_{\mathrm{dlog}}$ contains statements and witnesses corresponding to non-trivial discrete logarithm over the CRS, i.e., either the witness is $\mathbb{w} = z$ such that $Z = zG_1$ or $\mathbb{w} = \boldsymbol{v} \neq \mathbf{0}$ such that $0_{\mathbb{G}_1} = \sum_i v_i H_{\mathsf{trunc},i}$.

Lemma 5.3. *Let $N_{\mathsf{trunc}} = 2(n + n' + 1) + \ell_{\mathsf{cnt}} + 2$. The proof system $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ is correct, $(2N_{\mathsf{trunc}} + 1, N_{\mathsf{trunc}} + 11, 2, 2, k_1, \ldots, k_\mu)$-special sound for the relaxed relation $\widetilde{\mathcal{R}}_{\mathsf{trunc}}$, where $k_i = 3$ for all $i \in [\mu]$ with $\mu = \lceil \log(2N_{\mathsf{trunc}} + 7) \rceil$, and is perfect zero-knowledge.*

Proof. Note first that $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$ is correct, $(2N_{\mathsf{trunc}} + 1, N_{\mathsf{trunc}} + 11, 2, 2, k_1, \ldots, k_\mu)$-special sound, and special honest verifier zero-knowledge by the results in [AC20].

Correctness of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ then follows from correctness of $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$.

Zero-knowledge is based on a simulator Sim where (1) $\mathsf{Sim}_{\mathsf{Setup}}$ generates the CRS honestly except that Z is sampled with a trapdoor $z \in \mathbb{Z}_p$, and (2) $\mathsf{Sim}_{\mathsf{P}}(\mathsf{crs}, \mathsf{td}, \mathbb{x})$ first samples the challenges $\boldsymbol{c}' = (c_{1,0}, \ldots, c_{\mu+4,0})$, runs the HVZK simulator of $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$ on $(\mathsf{crs}, \mathbb{x}, \boldsymbol{c}')$, samples $R_i \leftarrow r_i G_1$ with $r_i \leftarrow\!\!\$\ \mathbb{Z}_p$ for $i \in [\mu + 4]$, and when the adversary sends a challenge c_i, compute $s_i \leftarrow r_i + (c_i - c_{i,0})z$. Statistical ZK then follows by HVZK of $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$ and that $(R_i, c_{i,1}, s_i)$ are identically distributed to the one in the actual protocol.

For **special soundness**, observe that for a $(2N_{\mathsf{trunc}} + 1, N_{\mathsf{trunc}} + 11, 2, 2, k_1, \ldots, k_\mu)$-tree of transcripts, if at some root of a subtree at level i the next level transcript contains distinct $c_{i,1}$ values along with s_i

such that $s_i G_1 - c_{i,1} Z = R_i$ (note that R_i is contained in the root of the subtree), then an extractor can extract the discrete log z. Otherwise, the subtree will contain identical $c_{i,1}$ meaning $c_{i,0}$ will be distinct as the challenges c_i are distinct. Therefore, we instead have a $(2N_{\mathsf{trunc}}+1, N_{\mathsf{trunc}}+11, 2, 2, k_1, \ldots, k_\mu)$-tree of $\Pi^{\mathsf{hvzk}}_{\mathsf{trunc}}$ transcripts instead, where its extractor can obtain a witness $\mathbb{w}$ of $\mathcal{R}_{\mathsf{trunc}}$ or some non-trivial discrete logarithm of $\boldsymbol{H}_{\mathsf{trunc}}$. Thus, proving special-soundness of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$. □

Next, we establish the security of the non-interactive proof Π_{trunc} in the following lemma. Soundness (when $\mathsf{crs}_{\mathsf{GS}}$ is generated in *binding mode*, i.e., via $\mathsf{GS.Setup}_{\mathsf{bind}}$) follows from the result of [AFK22]. If the CRS is not binding, any commitment can be opened to any openings, so any statements will always be in the language. Moreover, the reduction to DL is only *expected time* due to the resulting rewinding extractor from [AFK22].

Lemma 5.4. *Let* GGen *be a group generator outputting groups of prime-order* $p = p(\lambda)$*. Also, let* $n = \lfloor \log p \rfloor$, $n' = \lceil \log(p - 2^n) \rceil$ *and* $N_{\mathsf{trunc}} = 2(n + n' + 1) + \ell_{\mathsf{cnt}} + 2 = O(\lambda)$*. The proof system* Π_{trunc} *is correct, perfect zero-knowledge (in standard model), and when* $\mathsf{crs}_{\mathsf{GS}}$ *is generated in binding mode, sound for the language induced by the relation* $\mathcal{R}_{\mathsf{trunc}}$ *where for any adversary* $\mathcal{A}$ *running in time* $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$ *making at most* $Q_{\mathsf{H}} = Q_{\mathsf{H}}(\lambda)$ *queries to the random oracle* $\mathsf{H}_{\mathsf{trunc}}$*, there exists an adversary* $\mathcal{B}$ *playing the* DL *game running in expected time* $O(N^4_{\mathsf{trunc}}) Q_{\mathsf{H}} t_{\mathcal{A}}$ *such that*

$$\mathsf{Adv}^{\mathit{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}, \lambda) + (Q_{\mathsf{H}} + 1)\varepsilon + 1/p \,,$$

where $\varepsilon = \mathbf{Er}(2N_{\mathsf{trunc}} + 1, N_{\mathsf{trunc}} + 11, 2, 2, k_1, \ldots, k_\mu; p - N_{\mathsf{trunc}}, p, \ldots, p)$ *with* $\mathbf{Er}(k_1, \ldots, k_\mu; N_1, \ldots, N_\mu) = 1 - \prod_{i=1}^{\mu} (1 - \frac{k_i - 1}{N_i})$.

Proof (of Lemma 5.4). **Correctness** follows from correctness of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$.

Statistical zero-knowledge follows from statistical/perfect zero-knowledge of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$. In particular, we can go through the following sequence of games.

Game $\mathbf{G}^{\mathcal{A}}_0(\lambda)$. In this game, the CRS is generated from $\Pi_{\mathsf{trunc}}.\mathsf{Setup}$ algorithm and the adversary is given access to the oracle $\mathcal{O}_0$ which on input $(\mathbb{x}, \mathbb{w})$ return the proof π_{trunc} computed from $\Pi_{\mathsf{trunc}}.\mathsf{P}$ if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{trunc}}$.

Game $\mathbf{G}^{\mathcal{A}}_1(\lambda)$. In this game, the CRS is generated from $\mathsf{Sim}_{\mathsf{Setup}}$ of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ which generates crs along with a trapdoor td. By zero-knowledge of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ in Lemma 5.3, we have that

$$\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_1(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_0(\lambda) = 1] \,.$$

Game $\mathbf{G}^{\mathcal{A}}_2(\lambda)$. In this game, we now compute π_{trunc} in each oracle query from $\mathsf{Sim}_{\mathsf{P}}$ of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$. In particular, the oracle run the simulator and hash the output in each round to get the challenge for the next round. By zero-knowledge of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ in Lemma 5.3, we have that

$$\mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_2(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}^{\mathcal{A}}_1(\lambda) = 1] \,.$$

Hence, Π_{trunc} is perfect zero-knowledge.

For **soundness**, we consider an adversary $\mathcal{A}$ playing the soundness game. In particular, $\mathcal{A}$ takes as input $\mathsf{crs} = (\mathsf{crs}_{\mathsf{GS}}, \boldsymbol{H}_{\mathsf{trunc}}, Z)$ with $\mathsf{crs}_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{GS.Setup}_{\mathsf{bind}}(\mathsf{par})$ and $\boldsymbol{H}_{\mathsf{trunc}} \leftarrow\!\!\$\ \mathbb{G}_1^{2N_{\mathsf{trunc}}+7}, Z \leftarrow\!\!\$\ \mathbb{G}_1$ and returns a pair of statement and valid proof $(\mathbb{x}, \pi)$. Let Good denote the event where $\mathcal{A}$ wins in the soundness game, i.e., it outputs $\mathbb{x} = (\mathsf{com}_{\mathsf{cnt}}, \mathsf{com}_{y_1}, \mathsf{com}_{y_2}, \mathsf{com}_{\overline{y}_1}, \mathsf{com}_{\overline{y}_2})$ and π_{trunc} which verifies, but the committed values (which are statistically bound to the commitment) $(\mathsf{cnt}, y_1, y_2, \overline{y}_1, \overline{y}_2)$ such that $\overline{y}_j \not\equiv y_j \pmod{2^m}$ or $y_j \notin [0, 2^m - 1]$ for some $j \in [2]$ or $\mathsf{cnt} \notin [0, 2^{\ell_{\mathsf{cnt}}}]$. Note that $\mathsf{Pr}[\mathsf{Good}] = \mathsf{Adv}^{\mathsf{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{A}, \lambda)$.

To analyze $\mathsf{Pr}[\mathsf{Good}]$, we will employ the rewinding extractor of Attema et al. [AFK22], which we denote $\mathsf{Ext}_{\mathsf{AFK}}$. Our reduction $\mathcal{B}$ playing the DL game does the following:

- On input $(\mathsf{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}), G_1, X)$, $\mathcal{B}$ samples $\boldsymbol{\alpha}, \boldsymbol{\beta} \leftarrow\!\!\$\ \mathbb{Z}_p^{2N_{\mathsf{trunc}}+7}, a, b \leftarrow\!\!\$\ \mathbb{Z}_p$. Set $\boldsymbol{H}_{\mathsf{trunc}} \leftarrow \boldsymbol{\alpha} G_1 + \boldsymbol{\beta} X$ and $Z \leftarrow a G_1 + b X$. Also, sample $\mathsf{crs}_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{GS.Setup}_{\mathsf{bind}}(\mathsf{par})$. Set $\mathsf{crs} = (\mathsf{crs}_{\mathsf{GS}}, \boldsymbol{H}_{\mathsf{trunc}}, Z)$.
- Run $(\mathbb{x}, \pi, \mathbb{w}) \leftarrow\!\!\$\ \mathsf{Ext}^{\mathcal{A}}_{\mathsf{AFK}}(\mathsf{crs}, \rho_{\mathcal{A}})$.

- If $(\mathbb{x}, \mathbb{w}) \notin \widetilde{\mathcal{R}}_{\mathsf{trunc}}$, abort. Next, if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{trunc}}$, it is of the following form (1) If $\mathbb{w} = z$ where $zG_1 = Z$, then return $(z-a)/b$. (2) If $\mathbb{w} = \boldsymbol{v}'$ such that $v'_0 + \sum_{i=1}^{2N_{\mathsf{trunc}}+7} v'_i H_{\mathsf{trunc},i} = 0_{\mathbb{G}_1}$, then return $-\sum_{i=1}^{2N_{\mathsf{trunc}}+7} v'_i \alpha_i / (1 + \sum_{i=1}^{2N_{\mathsf{trunc}}+7} v'_i \beta_i)$. (Both of these assuming we divide with non-zero element, which occurs with probability at most $1/p$ as b and $\boldsymbol{\beta}$ are hidden from the view of $\mathcal{A}$.)

Note that $\mathcal{B}$ runs $\mathcal{A}$ at most $(Q+1)\cdot T$ times in expectation where $T = 4 \cdot 3^{\lceil \log(2N_{\mathsf{trunc}}+7) \rceil}(2N_{\mathsf{trunc}}+1)(N_{\mathsf{trunc}}+11) = O(N_{\mathsf{trunc}}^4)$. Let Bad be the event that $\mathcal{A}$ outputs a valid $(\mathbb{x}, \pi_{\mathsf{trunc}})$ but does not succeed in the soundness game and $\mathsf{Succ}_{\mathsf{Ext}}$ be the event that $\mathsf{Ext}_{\mathsf{AFK}}$ successfully outputs $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \widetilde{\mathcal{R}}_{\mathsf{trunc}}$ (in the relaxed relation). Then, by adaptive knowledge extraction of Π_{trunc} which follows from special soundness of $\Pi^{\mathsf{zk}}_{\mathsf{trunc}}$ and [AFK22, Proposition 2 and Theorem 4], we have that with ε as defined in the statement

$$\mathsf{Pr}\left[\mathsf{Succ}_{\mathsf{Ext}}\right] \geqslant \frac{\mathsf{Pr}[\mathsf{Good} \vee \mathsf{Bad}] - (Q+1)\varepsilon}{1-\varepsilon} \geqslant \mathsf{Pr}[\mathsf{Good} \vee \mathsf{Bad}] - (Q+1)\varepsilon \,,$$

with the second inequality following from $1-\varepsilon \leqslant 1$. Note that Good and Bad are disjoint by definition, so $\mathsf{Pr}[\mathsf{Good} \vee \mathsf{Bad}] = \mathsf{Pr}[\mathsf{Good}] + \mathsf{Pr}[\mathsf{Bad}]$. Then, notice that when $\mathsf{Ext}_{\mathsf{AFK}}$ succeeds and $\mathsf{Pr}[\mathsf{Bad}]$ does not occur, $\mathcal{B}$ wins the DL game except with probability $1/p$. Hence

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}, \lambda) &\geqslant \mathsf{Pr}[\mathsf{Succ}_{\mathsf{Ext}}] - \mathsf{Pr}[\mathsf{Bad}] - \frac{1}{p} \\
&\geqslant \mathsf{Pr}[\mathsf{Good}] - \frac{1}{p} - (Q+1)\varepsilon \\
&= \mathsf{Adv}^{\mathsf{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{A}, \lambda) - \frac{1}{p} - (Q+1)\varepsilon \,,
\end{aligned}$$

concluding the proof. □

5.3 Discussion

Efficiency. For efficiency, we instantiate the SPS scheme with Jutla-Roy [JR17], which is secure from the SXDH assumption with signatures of $5\mathbb{G}_1 + 1\mathbb{G}_2$ elements and verification of 3 pairing-product equations. We also require our group choice to have prime order p such that $p-1$ has a power-of-two divisor; specifically, BLS12-381 [Bow17] has power-of-two divisors as large as 2^{32}. Now, we consider the following asymptotic efficiency guarantees for each algorithms.

ISSUANCE PROTOCOL. The *user-side* algorithms is mostly dominated by sampling the function key and computing the commitments C along with the precomputed openings to $C_{T,j,i}$. This precomputation takes $O(k \log^2 k)$ group exponentiation. The *issuer-side*, however, only needs to compute $O(1)$ group exponentiations for the signing and committing to $\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1$. In terms of bandwidth, the user sends $12\mathbb{G}_1$ elements, while the issuer sends $5\mathbb{G}_1 + 1\mathbb{G}_2$ elements and 4 scalars for $\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1$. This is in total ≈1KB.

SHOWING. Showing algorithm's runtime is dominated by $O(N_{\mathsf{trunc}}) = O(\lambda)$ group exponentiations to compute the proof π_{trunc} and the openings to $C_{f,j}, C_{g,i}$. Other operations only take constant number of group exponentiation.

VERIFICATION. The verifier also needs to make $O(N_{\mathsf{trunc}}) = O(\lambda)$ group exponentiations to verify π_{trunc}, which dominated its computational cost.

TOKEN SIZE. Each token contains asymptotically $O(\log \lambda)$ group elements and scalars. Concretely, we consider $p \approx 2^{256}$ and $\ell_{\mathsf{cnt}} \approx 2^{20}$, resulting in $196\mathbb{G}_1 + 110\mathbb{G}_2 + 101\mathbb{Z}_p$ elements (≈ 23KBs with BLS12-381 curve), consisting of:

(a) The serial number and double spending equation consisting of 3 scalars.
(b) Groth-Sahai commitments to:
  (1) Commitments C, C_γ,
  (2) Signature σ,
  (3) Intermediate values $y_j, \overline{y}_j, z_i, t_{j,i}$, and

(4) Corresponding openings to the commitments.

These consist of $86\mathbb{G}_1 + 26\mathbb{G}_2$ elements.

(c) Groth-Sahai proof π_{GS} for the pairing-product equations (R.2) - (R.8), totaling 21 equations. The proof then contains $84(\mathbb{G}_1 + \mathbb{G}_2)$ elements. Note that this is a rough upper-bound as we can optimize further by using shorter proofs for multi-scalar equations on some of these as well.

(d) The proof π_{lin} consisting of 63 scalars.

(e) The proof π_{trunc}. This is a Fiat-Shamir compiled proof from a 2μ+1-move protocol where $\mu = \lceil \log(2N_{\mathsf{trunc}} + 7) \rceil + 3 = O(\log \lambda)$ with each move sending constant number of group elements and scalars. Thus, this consists of $O(\log \lambda)$ group elements. In total, this consists of $26\mathbb{G}_1 + 35\mathbb{Z}_p$ elements.

Dispenser size. The dispenser contains $O(k)$ group elements and scalars:

(a) The key key to the function F_{key}: $48k$ scalars in $\mathbb{Z}_p$ from the tables $T_{j,i} \in \mathbb{Z}_p^{8k}$ and $5d \approx 5\lambda$ coefficients of f_j, g_i.

(b) The commitments, signature, and γ: $17\mathbb{G}_1 + 1\mathbb{G}_2 + 4\mathbb{Z}_p$ elements.

(c) The precomputed openings to $\mathsf{VC}_{\mathsf{KZG}}$ commitments $C_{T,j,i}$: $48k\mathbb{G}_1$ elements.

(d) The states $\rho_{f,j}, \rho_{g,i}$ for computing openings to $C_{f,j}, C_{g,i}$: $5d\mathbb{Z}_p$ elements.

(e) Data structure for keeping track of counters keeps at most k counters.

Comparison with lattice-based alternatives. To the best of our knowledge, there does not exist any practical "context-based rate-limited" anonymous tokens from lattices. State-of-the-art for related primitives such as blind signatures (used as one-time tokens) and (non-rate-limiting) anonymous credentials have worse token sizes. For blind signatures [JS25], the token size and issuance bandwidth are ≈40KBs and ≈60KBs, resp. For anonymous credentials/signatures with efficient protocols [AGJ+24], the token size and issuance bandwidth are ≈80KBs and ≈45KBs, resp. One could also apply the CHKLM framework to lattice-based credentials—but with overhead costs depending on the PRF of choice. A recent work [BGY25] proposed a batched blind signature scheme with post-quantum security with the aforementioned approach and relying on the Crypto Dark Matter PRF [BIP+18]. Their approach has ≈0.68KBs issuance communication and ≈300KBs signature size. Compared to all of the above, our token size is smaller, and we achieve everlasting anonymity, instead of computational. Our issuance communication is comparable to [BGY25], while much better than the others.

Possible extension. Remark 3.1 informally discussed alternative security notion providing rate-limiting property with respect to k (i.e, users should not spend more than k tokens, otherwise they will be identified). Here, we sketch possible modifications to our construction to achieve such properties. Taking ideas from the e-cash construction of [CHL05], one can have the user additionally commit to another (statistically) pseudorandom function keys, later evaluated only on inputs in the range $[0, k-1]$. (This can also be a random vector of length k.)In particular, the user will now have two serial numbers and two double spending values (one for detecting double-spending per context and one for overall usage).

Potential areas for improvement. As the readers might have suspected, the efficiency of the scheme, although better than lattice-based alternatives, is still far from practical. A possible efficiency optimization could be achieved if one proves the security in idealized group models. Potentially, this could result in a tighter security proofs and reduce the token size by removing or simplifying the proofs of knowledge with more efficient alternatives.

6 Security of EARLT

In this section, we state the security guarantees for our $\mathsf{EARLT} = \mathsf{EARLT}[\mathsf{GGen}]$ scheme along with the proofs. As a short summary, with k, N, d as defined in Figure 9, we achieve (1) everlasting anonymity (Section 6.1), (2) unforgeability (Section 6.2) assuming the random oracle model (ROM), SXDH, and $8k$-SDH assumption, (3) linkability (Section 6.3) assuming the ROM, SXDH, $8k$-SDH and d-ARSDH assumptions, (4) exculpability (Section 6.4) assuming DLOG assumption.

6.1 Anonymity

Anonymity follows from statistical pseudorandomness of $\mathcal{F}$, perfect zero-knowledge of $\mathsf{GS}, \Pi_{\mathsf{lin}}$, and Π_{trunc}, and statistically hiding of $\mathsf{KZG}_{\mathsf{Ped}}$ and $\mathsf{VC}_{\mathsf{KZG}}$.

Theorem 6.1 (Anonymity of EARLT). *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order $p = p(\lambda)$,* $\mathsf{EARLT} = \mathsf{EARLT}[\mathsf{GGen}]$*, and $k = k(\lambda), N = N(\lambda), d = d(\lambda)$ be integers. Let $\mathsf{Sim}_{\mathsf{lin}}, \mathsf{Sim}_{\mathsf{trunc}}$ be simulators for Π_{lin} and Π_{trunc}. There exists a simulator* Sim *such that for any unbounded adversary $\mathcal{A}$ making at most $Q_{\mathrm{U}} = Q_{\mathrm{U}}(\lambda)$ queries to the oracle* U*, there exists adversaries $\mathcal{B}_{\mathcal{F}}$, $\mathcal{B}_{\mathsf{lin}}$, $\mathcal{B}_{\mathsf{trunc}}$, and $\mathcal{B}_{\mathsf{hide}}$ such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{EARLT},k,N,\mathsf{Sim}}(\mathcal{A},\lambda) \leqslant\ & \mathsf{Adv}^{zk}_{\Pi_{\mathsf{lin}},\mathsf{Sim}_{\mathsf{lin}}}(\mathcal{B}_{\mathsf{lin}},\lambda) + \mathsf{Adv}^{zk}_{\Pi_{\mathsf{trunc}},\mathsf{Sim}_{\mathsf{trunc}}}(\mathcal{B}_{\mathsf{trunc}},\lambda) \\ & + Q_{\mathrm{U}} \cdot \left(\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{B}_{\mathcal{F}},\lambda) + \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{VC}_{\mathsf{KZG}}}(\mathcal{B}_{\mathsf{hide}},\lambda)\right) .\end{aligned}$$

Then, the following corollary stating the concrete security bound for anonymity follows from Lemmas 4.3, 4.4, 5.1 and 5.4.

Corollary 6.2. *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order $p = p(\lambda)$,* $\mathsf{EARLT} = \mathsf{EARLT}[\mathsf{GGen}]$*, and $k = k(\lambda), N = N(\lambda), d = d(\lambda)$ be integers. There exists a simulator* Sim *such that for any unbounded adversary $\mathcal{A}$ making at most $Q_{\mathrm{U}} = Q_{\mathrm{U}}(\lambda)$ queries to the oracle* U*, we have that*

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{EARLT},k,N,\mathsf{Sim}}(\mathcal{A},\lambda) \leqslant Q_{\mathrm{U}} \cdot \left(\frac{k}{2^{d/2-6}} + \frac{8k}{p}\right) .$$

Proof (of Theorem 6.1). Let $\mathsf{Sim}_{\mathsf{GS}} = (\mathsf{Sim}_{\mathsf{GS,Setup}}, \mathsf{Sim}_{\mathsf{GS,Com}}, \mathsf{Sim}_{\mathsf{GS,P}})$, $\mathsf{Sim}_{\mathsf{lin}} = (\mathsf{Sim}_{\mathsf{lin,Setup}}, \mathsf{Sim}_{\mathsf{lin,P}})$, $\mathsf{Sim}_{\mathsf{trunc}} = (\mathsf{Sim}_{\mathsf{trunc,Setup}}, \mathsf{Sim}_{\mathsf{trunc,P}})$ be the simulators for the zero-knowledge properties of GS, Π_{lin} and Π_{trunc} respectively.

We now define the simulator Sim for our EARLT scheme as follows:

Setup $\mathsf{Sim}_{\mathsf{Setup}}(1^\lambda)$:

- $\mathsf{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda)$
- $\mathsf{crs}_{\mathsf{KZG}} \leftarrow\!\!\$\ \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Setup}(\mathsf{par}, d)$
- $\mathsf{crs}_{\mathsf{VC}} \leftarrow\!\!\$\ \mathsf{VC}_{\mathsf{KZG}}.\mathsf{Setup}(\mathsf{par}, S)$
- $(\mathsf{crs}_{\mathsf{GS}}, \mathsf{td}_{\mathsf{GS}}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{GS,Setup}}(\mathsf{par})$
- $(\mathsf{crs}_{\mathsf{lin}}, \mathsf{td}_{\mathsf{lin}}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{lin,Setup}}(\mathsf{crs}_{\mathsf{GS}})$
- $(\mathsf{crs}_{\mathsf{trunc}}, \mathsf{td}_{\mathsf{trunc}}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{trunc,Setup}}(\mathsf{crs}_{\mathsf{GS}})$.
- Return $(\mathsf{crs} = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}), \mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{crs}_{\mathsf{lin}}, \mathsf{crs}_{\mathsf{trunc}}, (H_i)_{i\in[4]}, \mathsf{td}' = (\mathsf{td}_{\mathsf{GS}}, \mathsf{td}_{\mathsf{lin}}, \mathsf{td}_{\mathsf{trunc}}))$.

User oracle $\mathsf{Sim}_{\mathrm{U}}$:

- For U_1, computes C as $C_{f,j}, C_{g,i} \leftarrow\!\!\$\ \mathsf{KZG.Com}(\mathsf{crs}_{\mathsf{KZG}}, 0)$ and $C_{T,j,i} \leftarrow\!\!\$\ \mathsf{VC}_{\mathsf{KZG}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{VC}}, \mathbf{0})$ for $j \in [2], i \in [3]$.
- For U_2, if the signature does not verify with respect to the (possibly) malicious issuer's secret key, abort. Otherwise, do nothing.

Showing oracle $\mathsf{Sim}_{\mathsf{Show}}(\mathsf{crs}, \mathsf{td}' = (\mathsf{td}, z), \mathsf{ctxt}, r)$: Return $(\mathsf{sn}, \tau = (\mathsf{dbsp}, \mathsf{com}, \pi = (\pi_{\mathsf{GS}}, \pi_{\mathsf{lin}}, \pi_{\mathsf{trunc}})))$ where

- $\mathsf{sn} \leftarrow\!\!\$\ \mathbb{Z}_p^2, \mathsf{dbsp} \leftarrow\!\!\$\ \mathbb{Z}_p$
- $(\mathsf{com}, \rho) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{GS,Com}}(\mathsf{td})$
- Set the statements as the algorithm EARLT.Show would.
  - $\mathbb{x}_{\mathsf{GS}} = (\mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{VC}}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{pk}_{\mathsf{I}}, \mathsf{ctxt}, \mathsf{sn}, \mathsf{dbsp}, r, \mathsf{com})$
  - $\mathbb{x}_{\mathsf{lin}} \leftarrow ((\mathsf{com}_{y_j}, \mathsf{com}_{\beta'_{f,j}})_{j,\in[2]}, (\mathsf{com}_{t_{1,i}}, \mathsf{com}_{t_{2,i}}, \mathsf{com}_{z_i}, \mathsf{com}_{\beta'_{g,i}})_{i\in[3]}, \mathsf{com}_{\gamma_0}, \mathsf{com}_{\gamma_1})$
  - $\mathbb{x}_{\mathsf{trunc}} = (\mathsf{com}_{\mathsf{cnt}}, (\mathsf{com}_{y_j}, \mathsf{com}_{\overline{y}_j})_{j\in[2]})$.
- Compute $\pi_{\mathsf{GS}} \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{GS,P}}(\mathsf{td}_{\mathsf{GS}}, \mathbb{x}_{\mathsf{GS}}, \rho)$, $\pi_{\mathsf{lin}} \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{lin,P}}(\mathsf{td}_{\mathsf{lin}}, \mathbb{x}_{\mathsf{lin}})$, $\pi_{\mathsf{trunc}} \leftarrow\!\!\$\ \mathsf{Sim}_{\Pi_{\mathsf{trunc}}}(\mathsf{td}_{\mathsf{trunc}}, \mathbb{x}_{\mathsf{trunc}})$.

We consider the following sequence of games.

Game $\mathbf{G}_0^{\mathcal{A}}(\lambda)$. This is exactly the $\mathrm{ANON}^{\mathcal{A}}_{\mathsf{EARLT},k,N,0}(\lambda)$ game.

Game $\mathbf{G}_1^{\mathcal{A}}(\lambda)$. This game samples the CRS with $(\mathsf{crs}_{\mathsf{GS}}, \mathsf{td}_{\mathsf{GS}}) \leftarrow_{\$} \mathsf{Sim}_{\mathsf{GS},\mathsf{Setup}}((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}))$. Note that since Setup generates $\mathsf{crs}_{\mathsf{GS}}$ in hiding mode, the $\mathsf{crs}_{\mathsf{GS}}$ is distributed as in $\mathbf{G}_0$. For Π_{lin} and Π_{trunc}, we have that the distributions of Setup and $\mathsf{Sim}_{\mathsf{Setup}}$ are identical. Hence,

$$\mathsf{Pr}[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}_0^{\mathcal{A}}(\lambda) = 1] \,.$$

Game $\mathbf{G}_2^{\mathcal{A}}(\lambda)$. This game now computes the proofs $\pi_{\mathsf{lin}}, \pi_{\mathsf{trunc}}$ by using the corresponding zero-knowledge simulator. In particular, the game samples $\mathsf{crs}_{\mathsf{lin}}$ and $\mathsf{crs}_{\mathsf{trunc}}$ by running $(\mathsf{crs}_{\mathsf{lin}}, \mathsf{td}_{\mathsf{lin}}) \leftarrow_{\$} \mathsf{Sim}_{\mathsf{lin},\mathsf{Setup}}(\mathsf{crs}_{\mathsf{GS}})$ and $(\mathsf{crs}_{\mathsf{trunc}}, \mathsf{td}_{\mathsf{trunc}}) \leftarrow_{\$} \mathsf{Sim}_{\mathsf{trunc},\mathsf{Setup}}(\mathsf{crs}_{\mathsf{GS}})$, and in each SHOW query computes $\pi_{\mathsf{lin}} \leftarrow_{\$} \mathsf{Sim}_{\mathsf{lin},\mathsf{P}}(\mathsf{td}_{\mathsf{lin}}, \mathbb{x}_{\mathsf{lin}})$ and $\pi_{\mathsf{trunc}} \leftarrow_{\$} \mathsf{Sim}_{\mathsf{trunc},\mathsf{P}}(\mathsf{td}_{\mathsf{trunc}}, \mathbb{x}_{\mathsf{trunc}})$. By statistical zero-knowledge properties of $\Pi_{\mathsf{lin}}, \Pi_{\mathsf{trunc}}$, there exists adversaries $\mathcal{B}_{\mathsf{lin}}, \mathcal{B}_{\mathsf{trunc}}$ such that

$$|\mathsf{Pr}[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] - \mathsf{Pr}[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1]| \leqslant \mathsf{Adv}^{\mathsf{zk}}_{\Pi_{\mathsf{lin}},\mathsf{Sim}_{\mathsf{lin}}}(\mathcal{B}_{\mathsf{lin}}, \lambda) + \mathsf{Adv}^{\mathsf{zk}}_{\Pi_{\mathsf{trunc}},\mathsf{Sim}_{\mathsf{trunc}}}(\mathcal{B}_{\mathsf{trunc}}, \lambda) \,.$$

Game $\mathbf{G}_3^{\mathcal{A}}(\lambda)$. In the showing oracle SHOW, the game now generates the commitment com and the proof π_{GS} via the simulators $\mathsf{Sim}_{\mathsf{GS},\mathsf{Com}}$ and $\mathsf{Sim}_{\mathsf{GS},\mathsf{P}}$ while using the trapdoor $\mathsf{td}_{\mathsf{GS}}$. Since GS is perfect zero-knowledge in hiding mode, we have that

$$\mathsf{Pr}[\mathbf{G}_3^{\mathcal{A}}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] \,.$$

Game $\mathbf{G}_4^{\mathcal{A}}(\lambda)$. The user oracle in this game computes the commitments C with $C_{f,j}, C_{g,i} \leftarrow_{\$} \mathsf{KZG.Com}(\mathsf{crs}_{\mathsf{KZG}}, 0)$ for $j \in [2], i \in [3]$, and $C_{T,j,i} \leftarrow_{\$} \mathsf{VC}_{\mathsf{KZG}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{VC}}, \mathbf{0})$. Note that by the change in $\mathbf{G}_3$ the commitments are not opened anymore. Now, we can apply the perfectly hiding property of the KZG and statistically hiding of $\mathsf{VC}_{\mathsf{KZG}}$ commitments, so that there exists an adversary $\mathcal{B}_{\mathsf{hide}}$ such that

$$|\mathsf{Pr}[\mathbf{G}_4^{\mathcal{A}}(\lambda) = 1] - \mathsf{Pr}[\mathbf{G}_3^{\mathcal{A}}(\lambda) = 1]| \leqslant Q_{\mathrm{U}} \cdot \mathsf{Adv}^{\mathsf{hide}}_{\mathsf{VC}_{\mathsf{KZG}}}(\mathcal{B}_{\mathsf{hide}}, \lambda) \,.$$

Note that the bound follows from standard hybrid argument. Also, everything in this game except for the values $\mathsf{sn}, \mathsf{dbsp}$ are independent of the Pagh-Pagh function keys.

Game $\mathbf{G}_5^{\mathcal{A}}(\lambda)$. This game samples sn and dbsp as uniformly random elements in $\mathbb{Z}_p^2$ and $\mathbb{Z}_p$, respectively. Now, the showing does not depend on any component of the issuance protocol. This is exactly the game where everything is simulated. To finally argue the indistinguishability, notice that for each token dispenser generated in $\mathrm{U}_1, \mathrm{U}_2$, the adversary can ask for at most k showing per dispenser. Note that sn and dbsp are generated from $\mathcal{F}_{\mathsf{key}}$ in $\mathbf{G}_4$. By PRF-security of $\mathcal{F}$ against any unbounded adversary and a standard hybrid argument, there exists an adversary $\mathcal{B}_{\mathcal{F}}$ such that

$$|\mathsf{Pr}[\mathbf{G}_5^{\mathcal{A}}(\lambda) = 1] - \mathsf{Pr}[\mathbf{G}_4^{\mathcal{A}}(\lambda) = 1]| \leqslant Q_{\mathrm{U}} \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{B}_{\mathcal{F}}, \lambda) \,. \square$$

6.2 Unforgeability

Our unforgeability follows from the security of GS, Π_{trunc}, unforgeability of SPS, and the evaluation- and position-binding of $\mathsf{KZG}_{\mathsf{Ped}}$ and $\mathsf{VC}_{\mathsf{KZG}}$ commitments, respectively. Ultimately, unforgeability is implied by q-SDH and SXDH assumptions (with $q = 8k$).

Theorem 6.3 (Unforgeability of EARLT). *Let GGen be a bilinear group parameters generator outputting groups of prime-order $p = p(\lambda)$, $\mathsf{EARLT} = \mathsf{EARLT}[\mathsf{GGen}]$, and $k = k(\lambda), N = N(\lambda), d = d(\lambda)$ be integers. For any adversary $\mathcal{A}$ against the* UNF *game of EARLT running in time $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$ and making at most $Q_{\mathrm{Iss}} = Q_{\mathrm{Iss}}(\lambda), Q_{\mathsf{H}^\star} = Q_{\mathsf{H}^\star}(\lambda)$ queries to the issuance oracle and random oracle $\mathsf{H}^\star \in \{\mathsf{H}_{\mathsf{lin}}, \mathsf{H}_{\mathsf{trunc}}\}$, respectively, there exist adversaries $\mathcal{B}_{\mathsf{dist}}, \mathcal{B}_{\mathsf{SPS}}, \mathcal{B}_{\Pi}, \mathcal{B}_{\mathsf{ebind}}, \mathcal{B}_{\mathsf{pbind}}, \mathcal{B}_{\mathrm{dlog}}$ and $\mathcal{B}'_{\mathrm{dlog}}$ such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{EARLT},k,N}(\mathcal{A}, \lambda) \leqslant\ & \mathsf{Adv}^{\mathsf{dist}}_{\mathsf{GS},\mathsf{Ext}_{\mathsf{Setup}}}(\mathcal{B}_{\mathsf{dist}}, \lambda) + 8(Q_{\mathrm{Iss}} N + 1)\mathsf{Adv}^{\mathit{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{B}_{\Pi}, \lambda) \\ & + \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SPS}}(\mathcal{B}, \lambda) + 8(\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}'_{\mathrm{dlog}}, \lambda) + \\ & + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}}, \lambda) + \mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}', \lambda)) \,.\end{aligned}$$

Additionally, $\mathcal{B}_{\mathsf{dist}}, \mathcal{B}_{\mathsf{SPS}}$*, and* $\mathcal{B}_{\Pi}$ *run in time* $\approx t_{\mathcal{A}}$ *with* $\mathcal{B}_{\Pi}$ *making* $Q_{\mathsf{H}_{\mathsf{trunc}}}$ *queries to* $\mathsf{H}_{\mathsf{trunc}}$*, and* $\mathcal{B}_{\mathrm{dlog}}, \mathcal{B}'_{\mathrm{dlog}}, \mathcal{B}_{\mathsf{ebind}}$*, and* $\mathcal{B}_{\mathsf{pbind}}$ *run in time* $\approx \frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon}\ln(\frac{16}{\varepsilon})t_{\mathcal{A}}$ *where* $\varepsilon(\lambda) \geqslant \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{EARLT},k,N}(\mathcal{A},\lambda) - \mathsf{Adv}^{\mathsf{dist}}_{\mathsf{GS},\mathsf{Ext}_{\mathsf{Setup}}}(\mathcal{B}_{\mathsf{dist}},\lambda) - \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SPS}}(\mathcal{B},\lambda)$.

Proof (of Theorem 6.3). Let $\mathcal{A}$ be an adversary playing the unforgeability game of our EARLT scheme, who makes $Q_{\mathrm{Iss}}, Q_{\mathsf{H}_{\mathsf{lin}}}, Q_{\mathsf{H}_{\mathsf{trunc}}}$ queries to the oracle Iss and the random oracle $\mathsf{H}_{\mathsf{lin}}$ and $\mathsf{H}_{\mathsf{trunc}}$ (resp.), and at the end of the game outputs a context string ctxt, and $QN+1$ tuples $(r_k, \mathsf{sn}_k, \tau_k)_{i\in[QN+1]}$ where r_k is the nonce for which sn_k, τ_k verifies to. Note that each token τ_k can be parsed as $(\mathsf{dbsp}_k, \mathsf{com}_k, \pi_k = (\pi_{\mathsf{GS},k}, \pi_{\mathsf{lin},k}, \pi_{\mathsf{trunc},k}))$ for $k \in [QN+1]$. Without loss of generality (with a small increase in the query count), we assume that $\mathcal{A}$ makes the RO queries to made during verification of the tokens. Now, we consider the following sequence of games.

Game $\mathbf{G}_0^{\mathcal{A}}(\lambda)$**.** This game is identical to the $\mathrm{UNF}_{\mathsf{EARLT},k,N}$ game.

Game $\mathbf{G}_1^{\mathcal{A}}(\lambda)$**.** This game generates the CRS for GS, $\mathsf{crs}_{\mathsf{GS}}$ along with a trapdoor td using the extractor set up $\mathsf{Ext}_{\mathsf{Setup}}$ as defined in F-knowledge of GS. It is easy to see that there exists an adversary $\mathcal{B}_{\mathsf{dist}}$ playing the CRS indisintguishability game of $\mathsf{Ext}_{\mathsf{Setup}}$ such that

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

Game $\mathbf{G}_2^{\mathcal{A}}(\lambda)$**.** In this game, the game uses the extractor $\mathsf{Ext}_{\mathsf{P}}(\mathsf{td},\cdot)$ on

$$(\mathbb{x}_k = (\mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{pk}_{\mathsf{I}}, r_k, \mathsf{ctxt}_k, \mathsf{sn}_k, \mathsf{dbsp}_k, \mathsf{com}_k), \pi_{\mathsf{GS},k})$$

for each $k \in [QN+1]$ to get the witness $\widetilde{\mathbb{w}}^{(k)}$ of the relation $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ containing the following group elements (omitting the superscript $(\cdot)^{(k)}$ for readability) that satisfies Equation (1).

$$\begin{aligned}
&\widehat{\mathsf{cnt}}_1 \in \mathbb{G}_1, \widehat{\mathsf{cnt}}_2 \in \mathbb{G}_2,\\
&\mathsf{pk}_{\mathsf{User}} \in \mathbb{G}_1, C_\gamma \in \mathbb{G}_1,\\
&C = (C_{f,1}, C_{f,2}, (C_{g,i}, C_{T,1,i}, C_{T,2,i})_{i\in[3]}) \in \mathbb{G}_1^{11}, \sigma,\\
&(Y_j \in \mathbb{G}_1, \overline{Y}_{1,j} \in \mathbb{G}_1, \overline{Y}_{2,j} \in \mathbb{G}_2, \mathsf{open}_{f,j} = (B'_{f,j}, Q_{f,j}) \in \mathbb{G}_1^2)_{j\in[2]},\\
&(Z_i \in \mathbb{G}_1, \mathsf{open}_{g,i} = (B'_{g,i}, Q_{g,i}) \in \mathbb{G}_1^2, (\hat{T}_{j,i} \in \mathbb{G}_1, \mathsf{open}_{T,j,i} \in \mathbb{G}_1)_{j\in[2]})_{i\in[3]},\\
&\boldsymbol{\Gamma}_0, \boldsymbol{\Gamma}_1 \in \mathbb{G}_2^2
\end{aligned}$$

The game aborts if $(\mathbb{x}_k, \widetilde{\mathbb{w}}^{(k)}) \notin \widetilde{\mathcal{R}}_{\mathsf{tok}}$. By perfect F-knowledge of GS,

$$\mathsf{Pr}[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1]\ .$$

Now, consider the event that $\mathcal{A}$ wins in game $\mathbf{G}_2$. We consider the following cases:

- $\mathsf{Forge}_{\mathsf{SPS}}$: There exists some $k \in [QN+1]$ such that $\mathbb{w}^{(k)}$ contains valid $\sigma^{(k)}$ for $(\mathsf{pk}^{(k)}_{\mathsf{User}}, C^{(k)}, C^{(k)}_\gamma)$ which was not signed in any issuance oracle query. In this case, we can construct a reduction $\mathcal{B}_{\mathsf{SPS}}$ to the EUF-CMA property of SPS. The reduction would simulate the game $\mathbf{G}_2$ honestly, except that to reply to issuance oracle queries, the reduction queries its signing oracle in its unforgeability game. Then, when the event $\mathsf{Forge}_{\mathsf{SPS}}$ occurs, the reduction outputs the forgery $((\mathsf{pk}^{(k)}_{\mathsf{User}}, C^{(k)}, C^{(k)}_\gamma), \sigma^{(k)})$. Hence, there exists an adversary $\mathcal{B}_{\mathsf{SPS}}$ running in time roughly $t_{\mathcal{A}}$ such that $\mathsf{Pr}[\mathsf{Forge}_{\mathsf{SPS}}] \leqslant \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SPS}}(\mathcal{B}_{\mathsf{SPS}},\lambda)$.
- $\mathsf{BadForm}$: There exists some $k \in [QN+1]$ such that $\mathbb{w}^{(k)}$ contains $\widehat{\mathsf{cnt}}^{(k)}_1, (Y^{(k)}_j, \overline{Y}^{(k)}_{1,j})_{j\in[2]}$ such that one of the following is true: (a) $\mathrm{dlog}_{G_1}\widehat{\mathsf{cnt}}^{(k)}_1 \notin [0, N-1]$, (b) $\mathrm{dlog}_{G_1} Y^{(k)}_j \neq \mathrm{dlog}_{G_1}\overline{Y}^{(k)}_{1,j} \pmod{2^m}$, or (c) $\mathrm{dlog}_{G_1}\overline{Y}^{(k)}_{1,j} \notin [0, 2^m-1]$. This means that soundness of Π_{trunc} is broken. However, note that this particular event is inefficient to check, but we could still construct a reduction by guessing $k^* \in [QN+1]$ such that the event occurs on and output the corresponding proof. Hence, there exists an adversary $\mathcal{B}_{\Pi}$ against soundness of Π_{trunc} such that $\mathsf{Pr}[\mathsf{BadForm}] \leqslant (QN+1)\cdot\mathsf{Adv}^{\mathsf{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{B}_{\Pi},\lambda)$, and $\mathcal{B}_{\Pi}$ runs in time roughly that of $\mathcal{A}$ while making at most $Q_{\mathsf{H}_{\mathsf{trunc}}}$ queries to $\mathsf{H}_{\mathsf{trunc}}$.

AllSigned: All witnesses $\widetilde{\mathbb{w}}^{(k)}$ contain $(\mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$ which was queried to the issuance oracle and BadForm does not occur. Note that in this case, $\mathsf{cnt}^{(k)} = \mathrm{dlog}_{G_1} \widehat{\mathsf{cnt}}_1^{(k)}$ and $\overline{y}_{1,j}^{(k)} = \mathrm{dlog}_{G_1} Y_{1,j}^{(k)}$ can be efficiently computed, since N and 2^m are small. Then, by Pigeonhole's Principle (over $QN+1$ tokens), we have that there are $k_0 \neq k_1$ (for consistency, let (k_0, k_1) be the lexicographically first such pair) such that

$$(\mathsf{cnt}^{(k_b)}, \mathsf{pk}_{\mathsf{User}}^{(k_b)}, C^{(k_b)}, C_\gamma^{(k_b)})$$

are identical for $b \in \{0,1\}$. For readability, we denote the values $(\mathsf{sn}_{k_b}, \tau_{k_b} = (\mathsf{dbsp}_{k_b}, \mathsf{com}_{k_b}, \pi_{k_b}))$ along with the witness $\widetilde{\mathbb{w}}^{(k_b)}$ as $(\mathsf{sn}_b, \tau_b = (\mathsf{dbsp}_b, \mathsf{com}_b, \pi_b), \widetilde{\mathbb{w}}^{(b)})$ for $b \in \{0,1\}$. The values parsed from these values also follow suit.

Note that $\Pr[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] \leqslant \Pr[\mathsf{Forge}_{\mathsf{SPS}}] + \Pr[\mathsf{AllSigned}] + \Pr[\mathsf{BadForm}]$.

To bound $\Pr[\mathsf{AllSigned}]$, we first define a wrapper $\mathcal{A}'$ (given in Figure 16)which takes as input crs, the issuer's secret key sk_{I} and public key pk_{I}, and a trapdoor td for the extractor of GS, sampled from the same distribution as in $\mathbf{G}_2$. The adversary $\mathcal{A}'$ has access to the random oracle $\mathsf{H}_{\mathsf{lin}}$ making at most $Q_{\mathsf{H}_{\mathsf{lin}}}$ queries. It runs the adversary $\mathcal{A}$ as in $\mathbf{G}_2$ (sk_{I} is used to simulate the issuer's oracle, the $\mathsf{H}_{\mathsf{lin}}$ queries are replied by querying its own RO, and $\mathsf{H}_{\mathsf{trunc}}$ queries are simulated using its own random coins). At the end of the game, $\mathcal{A}'$ checks whether $\mathcal{A}$ wins the game and $\mathsf{Forge}_{\mathsf{SPS}}$ does not occur. If so, it returns $(\mathsf{ctxt}, (r_b, \mathsf{sn}_b, \tau_b, \widetilde{\mathbb{w}}^{(b)})_{b\in\{0,1\}})$ as defined in the event $\mathsf{AllSigned}$. Note that this does not rule out the event $\mathsf{BadForm}$, so $\Pr[\mathcal{A}'(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \mathsf{td}) \neq \bot] = \Pr[\mathsf{AllSigned} \vee \mathsf{BadForm}]$.

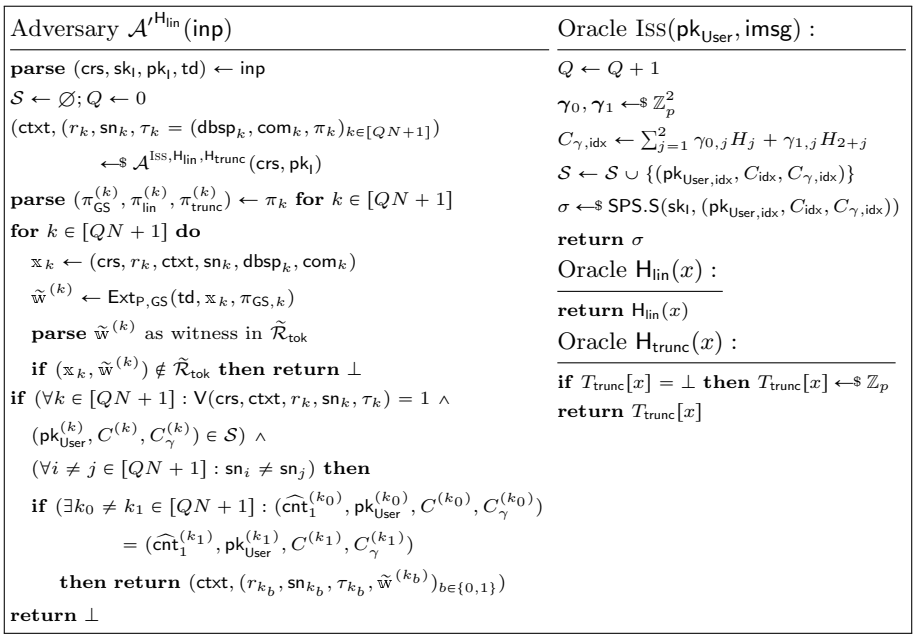

Fig. 16. Wrapper adversary $\mathcal{A}'$ for bounding $\Pr[\mathsf{AllSigned}]$

Now, we want to apply the extractor $\mathsf{Ext}_{\mathsf{lin}}$ from Lemma 5.1 with $L = 2$ on $\mathcal{A}'$ as follows. First, run the wrapper $\mathcal{A}'$ on input inp with the random coins $\rho_{\mathcal{A}'}$ and simulate the oracle $\mathsf{H}_{\mathsf{lin}}$ to $\mathcal{A}'$ (letting $\boldsymbol{h}$ be the RO outputs). Then, on the output $\mathsf{out} = (\mathsf{ctxt}, (r_b, \mathsf{sn}_b, \tau_b, \widetilde{\mathbb{w}}^{(b)})_{b\in\{0,1\}})$ of $\mathcal{A}'$, run the extractor $\mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}'}(\mathsf{inp}, \mathsf{out}, \boldsymbol{h}; \rho_{\mathcal{A}'})$ which returns the witnesses $\mathbb{w}_{\mathsf{lin}}^{(0)}, \mathbb{w}_{\mathsf{lin}}^{(1)}$, containing the scalar openings.

Denote the event Good as the one where the openings $\mathbb{w}_{\mathsf{lin}}^{(b)}$ defined as

$$\mathbb{w}_{\mathsf{lin}}^{(b)} = ((y_j^{(b)}, \mathsf{rand}_{y_j}^{(b)}, {\beta'_{f,j}}^{(b)}, \mathsf{rand}_{\beta'_{f,j}}^{(b)})_{j\in[2]},$$
$$(z_i^{(b)}, \mathsf{rand}_{z_i}^{(b)}, {\beta'_{g,i}}^{(b)}, \mathsf{rand}_{\beta'_{g,i}}^{(b)}, (t_{j,i}^{(b)}, \mathsf{rand}_{t_{j,i}}^{(b)})_{j\in[2]})_{i\in[3]}, \boldsymbol{\gamma}_0^{(b)}, \boldsymbol{\gamma}_1^{(b)}, \mathsf{rand}_{\boldsymbol{\gamma}_0}^{(b)}, \mathsf{rand}_{\boldsymbol{\gamma}_1}^{(b)})$$

satisfy $y_j^{(b)} = \overline{y}_j^{(b)} \pmod{2^m}$ for $b \in \{0,1\}, j \in [2]$ (i.e., $\mathsf{BadForm}$ does not occur with respect to the outputs of $\mathcal{A}'$). Then, there exists a DL adversary $\mathcal{B}_{\mathrm{dlog}}$ such that (by Lemma 5.1 and definition of $\mathsf{AllSigned}$)

$$\mathsf{Pr}[\mathsf{Good} \vee \mathsf{BadForm}] \geqslant \frac{\mathsf{Pr}[\mathsf{AllSigned} \vee \mathsf{BadForm}]}{8} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}}, \lambda)$$
$$\mathsf{Pr}[\mathsf{Good}] \geqslant \frac{\mathsf{Pr}[\mathsf{AllSigned}] - 7 \cdot \mathsf{Pr}[\mathsf{BadForm}]}{8} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}}, \lambda)\ .$$

The second line follows from $\mathsf{AllSigned}$ and $\mathsf{BadForm}$ being disjoint and a union bound on $\mathsf{Pr}[\mathsf{Good} \vee \mathsf{BadForm}]$. Note that $\mathsf{Ext}_{\mathsf{lin}}$ runs $\mathcal{A}'$ at most $32Q_{\mathsf{H}}/\varepsilon \ln(16/\varepsilon)$ times where $\varepsilon(\lambda) = \mathsf{Pr}[\mathsf{AllSigned} \vee \mathsf{BadForm}]$.

By the definition of $\mathcal{A}'$, if Good occurs, the output of $\mathcal{A}'$, $(\mathsf{sn}_b, \tau_b = (\mathsf{dbsp}_b, \mathsf{com}_b, \pi_b), \widetilde{\mathbb{w}}^{(b)})$ contains the same $\widehat{\mathsf{cnt}}_1^{(b)}, \mathsf{pk}_{\mathsf{User}}^{(b)}, C^{(b)}, C_\gamma^{(b)}$ (i.e., the same $\mathsf{cnt} = \mathsf{cnt}^{(b)} = \mathrm{dlog}_{G_1} \widehat{\mathsf{cnt}}_1^{(b)}$). Now, since that $\mathsf{sn}_0 \neq \mathsf{sn}_1$ (since $\mathcal{A}$ wins in the game), we consider the extracted values contained in $\mathbb{w}_{\mathsf{lin}}^{(b)}$.

Now, we consider the following cases:

- $y_j^{(0)} \neq y_j^{(1)}$ for some $j \in [2]$. In this case, since $Y_j^{(b)} = y_j^{(b)} G_1$ and ${B'_{f,j}}^{(b)} = {\beta'_{f,j}}^{(b)} H$ for $b \in \{0,1\}$, and by how $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ is defined

$$\mathsf{e}\left(C_{f,j}^{(b)} - Y_j^{(b)} - {B'_{f,j}}^{(b)}, G_2\right) = \mathsf{e}\left(Q_{f,j}^{(b)}, X_{2,1} - (\mathsf{cnt} + \mathsf{ctxt} \cdot 2^{\ell_{\mathsf{ctr}}})G_2\right)$$

  Since $C_{f,i}^{(0)} = C_{f,i}^{(1)}$ and the openings $({\beta'_{f,j}}^{(b)}, Q_{f,j}^{(b)})$ are at the same point $\mathsf{cnt} + \mathsf{ctxt} \cdot 2^{\ell_{\mathsf{ctr}}}$ but $y_j^{(0)} \neq y_j^{(1)}$, we break evaluation binding of $\mathsf{KZG}_{\mathsf{Ped}}$ commitment scheme.
- $z_i^{(0)} \neq z_i^{(1)}$ for some $i \in [2]$. Similarly to the case above, this also breaks binding of $\mathsf{KZG}_{\mathsf{Ped}}$ commitment scheme on the commitment $C_{g,i}$.
- $y_j^{(0)} = y_j^{(1)}$ for all $j \in [2]$ and $t_{1,i}^{(0)} \neq t_{1,i}^{(1)}$ or $t_{2,i}^{(0)} \neq t_{2,i}^{(1)}$ for some $i \in [2]$. Since $y_j^{(0)} = y_j^{(1)}$, we also have that $\overline{y}_j^{(0)} = \overline{y}_j^{(1)}$ (since $\overline{y}_j^{(b)} = y_j^{(b)} \pmod{2^m}$ and $\overline{y}_j^{(b)} \in [0, 2^m - 1]$), and accordingly with $t_{1,i}^{(0)} \neq t_{1,i}^{(1)}$ or $t_{2,i}^{(0)} \neq t_{2,i}^{(1)}$, we break position binding of $\mathsf{VC}_{\mathsf{KZG}}$ commitment $C_{T,1,i}$ or $C_{T,2,i}$.
- $\boldsymbol{\gamma}_0^{(0)} \neq \boldsymbol{\gamma}_0^{(1)}$ or $\boldsymbol{\gamma}_1^{(0)} \neq \boldsymbol{\gamma}_1^{(1)}$. Here, since $C_\gamma = C'_\gamma$, this breaks the discrete logarithm as it gives a non-trivial equation over the group elements H_1, H_2, H_3, H_4.

Note that if none of the above occurs, then $\mathsf{sn}_0 = \mathsf{sn}_1$ by how $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ is defined, which is a contradiction. Therefore, there exists an adversary $\mathcal{B}_{\mathsf{ebind}}, \mathcal{B}_{\mathsf{pbind}}$ and $\mathcal{B}'_{\mathrm{dlog}}$ playing the binding game of $\mathsf{KZG}_{\mathsf{Ped}}$ commitment and the DL game such that

$$\mathsf{Pr}[\mathsf{Good}] \leqslant \mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{ebind}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}'_{\mathrm{dlog}}, \lambda)\ ,$$

implying that

$$\mathsf{Pr}[\mathsf{AllSigned}] \leqslant 8(\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{ebind}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}}, \lambda) +$$
$$\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}'_{\mathrm{dlog}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}}, \lambda)) + 7\mathsf{Pr}[\mathsf{BadForm}]\ .$$

Combining with the bounds on $\mathsf{Pr}[\mathsf{Forge}_{\mathsf{SPS}}]$ and $\mathsf{Pr}[\mathsf{BadForm}]$, we conclude the proof. □

6.3 Linkability

Theorem 6.4 (Linkability of EARLT). *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order* $p = p(\lambda)$*,* $\mathsf{EARLT} = \mathsf{EARLT}[\mathsf{GGen}]$*, and* $k = k(\lambda), N = N(\lambda)$ *be integers. For any adversary* $\mathcal{A}$ *playing the* LINK *game of* EARLT *running in time* $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$ *and making at most* $Q_{\mathrm{Iss}} = Q_{\mathrm{Iss}}(\lambda)$*,* $Q_{\mathsf{H}^\star} = Q_{\mathsf{H}^\star}(\lambda)$ *queries to the issuance oracle and random oracles* $\mathsf{H}^\star \in \{\mathsf{H}_{\mathsf{lin}}, \mathsf{H}_{\mathsf{trunc}}\}$*, respectively, there exist adversaries* $\mathcal{B}_{\mathsf{dist}}, \mathcal{B}_{\mathsf{SPS}}, \mathcal{B}_{\mathrm{dlog},1}, \mathcal{B}_{\mathrm{dlog},2}, \mathcal{B}_{\mathrm{dlog},3}, \mathcal{B}_{\mathrm{dlog},4}, \mathcal{B}_{\mathsf{ebind}}, \mathcal{B}_{\mathsf{pbind}}, \mathcal{B}_{\mathsf{Coll,KZG}}, \mathcal{B}'_{\mathsf{Coll,KZG}}, \mathcal{B}_{\mathsf{Coll,VC}}, \mathcal{B}'_{\mathsf{Coll,VC}}$ *such that*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{link}}_{\mathsf{EARLT},k,N}(\mathcal{A},\lambda) \leqslant\ & \mathsf{Adv}^{\mathsf{dist}}_{\mathsf{GS},\mathsf{Ext}_{\mathsf{Setup}}}(\mathcal{B}_{\mathsf{dist}},\lambda) + \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SPS}}(\mathcal{B}_{\mathsf{SPS}},\lambda) + \frac{Q_{\mathrm{Iss}}^2}{p} + \frac{Q_{\mathrm{Iss}}^2}{2^{2\lambda}} \\
& + 16(Q_{\mathrm{Iss}}^2 + Q_{\mathrm{Iss}} + 1)\mathsf{Adv}^{\mathit{sound}}_{\mathit{\Pi}_{\mathsf{trunc}}}(\mathcal{B}_{\mathit{\Pi}},\lambda) + 16\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},2},\lambda) \\
& + 8(\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{ebind}},\lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}},\lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},1},\lambda)) \\
& + 8Q_{\mathrm{Iss}}(\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{Coll,KZG}},\lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{Coll,VC}},\lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3},\lambda)) \\
& + 8Q_{\mathrm{Iss}}^2(\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}'_{\mathsf{Coll,KZG}},\lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}'_{\mathsf{Coll,VC}},\lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},4},\lambda)) \ .
\end{aligned}$$

Additionally,

- $\mathcal{B}_{\mathsf{dist}}, \mathcal{B}_{\mathsf{SPS}}, \mathcal{B}_{\mathit{\Pi}}$ *runs in time* $\approx t_{\mathcal{A}}$ *with* $\mathcal{B}_{\mathit{\Pi}}$ *making at most* $Q_{\mathsf{H}_{\mathsf{trunc}}}$ *queries to* $\mathsf{H}_{\mathsf{trunc}}$
- $\mathcal{B}_{\mathsf{ebind}}, \mathcal{B}_{\mathsf{pbind}}, \mathcal{B}_{\mathrm{dlog},1}$ *all run in time* $\approx \frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_1(\lambda)}\ln(\frac{16}{\varepsilon_1(\lambda)})t_{\mathcal{A}}$
- $\mathcal{B}_{\mathrm{dlog},2}$ *runs in time* $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_2(\lambda)}\ln(\frac{16}{\varepsilon_2(\lambda)})t_{\mathcal{A}}$ *and* $\mathcal{B}_{\mathrm{dlog},3}$ *runs in time* $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_3(\lambda)}\ln(\frac{16}{\varepsilon_3(\lambda)})t_{\mathcal{A}}$
- $\mathcal{B}_{\mathsf{Coll,KZG}}$ *and* $\mathcal{B}_{\mathsf{Coll,VC}}$ *run in time* $\frac{256k\lambda Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_3\varepsilon_{\mathsf{SNColl}}}\ln(\frac{16}{\varepsilon_3})t_{\mathcal{A}}$*,*
- $\mathcal{B}_{\mathrm{dlog},4}$ *runs in time* $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_4(\lambda)}\ln(\frac{16}{\varepsilon_4(\lambda)})t_{\mathcal{A}}$*, and* $\mathcal{B}'_{\mathsf{Coll,KZG}}, \mathcal{B}'_{\mathsf{Coll,VC}}$ *run in time* $\frac{256k\lambda Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_4\varepsilon_{\mathsf{SNColl}_2}}\ln(\frac{16}{\varepsilon_4})t_{\mathcal{A}}$*,*

where the functions $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_{\mathsf{SNColl}}$*, and* $\varepsilon_{\mathsf{SNColl}_2}$ *are such that*

- $\varepsilon_1(\lambda) + \varepsilon_2(\lambda) + Q_{\mathrm{Iss}}\varepsilon_3(\lambda) + Q_{\mathrm{Iss}}^2\varepsilon_4(\lambda) \geqslant \mathsf{Adv}^{\mathsf{link}}_{\mathsf{EARLT},k,N}(\mathcal{A},\lambda) - \mathsf{Adv}^{\mathsf{dist}}_{\mathsf{GS},\mathsf{Ext}_{\mathsf{Setup}}}(\mathcal{B}_{\mathsf{dist}},\lambda) - \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SPS}}(\mathcal{B},\lambda)$
- $\varepsilon_{\mathsf{SNColl}}(\lambda) \geqslant \frac{\varepsilon_3(\lambda)}{8} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3},\lambda) - 2\mathsf{Adv}^{\mathit{sound}}_{\mathit{\Pi}_{\mathsf{trunc}}}(\mathcal{B}_{\mathit{\Pi}},\lambda)$
- $\varepsilon_{\mathsf{SNColl}_2}(\lambda) \geqslant \frac{\varepsilon_4(\lambda)}{8} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},4},\lambda) - 2\mathsf{Adv}^{\mathit{sound}}_{\mathit{\Pi}_{\mathsf{trunc}}}(\mathcal{B}_{\mathit{\Pi}},\lambda)$

We now provide more high-level idea for the proof. Following the sketch in Section 1.1, we want to bound the probability that the adversary can force a collision in the serial number either when the tokens come from the same dispenser or from different ones. Intuitively, if the function key is already fixed by the KZG commitments, the randomization factor γ_0, γ_1 introduced into the computation of the serial number can prevent such event. Indeed, such assumption does not hold statistically, so we have to somehow give a reduction to the binding property of the commitments. Our approach is to observe that if the adversary is able to force the collision for multiple random γ_0, γ_1 (which is sent after the commitments are determined in the issuance oracle), we can reduce to degree-binding (see Section 2) of $\mathsf{KZG}_{\mathsf{Ped}}$ and position-binding of $\mathsf{VC}_{\mathsf{KZG}}$. Essentially, our reduction will *rewind* the adversary enough times with different randomness γ_0, γ_1, so that eventually the adversary becomes inconsistent with the committed polynomial or the committed vector, allowing us to break degree-binding or position-binding of $\mathsf{KZG}_{\mathsf{Ped}}$ or $\mathsf{VC}_{\mathsf{KZG}}$ commitments.

Proof (of Theorem 6.4). Let $\mathcal{A}$ be an adversary playing the linkability game of our EARLT scheme, who makes $Q_{\mathrm{Iss}}, Q_{\mathsf{H}}$ queries to the oracle Iss and the random oracle H (resp.), and at the end of the game outputs ctxt and 2 tuples $(r_k, \mathsf{sn}_k, \tau_k)_{k\in\{0,1\}}$ where r_k is the corresponding nonce value which the token verifies. Note that each token τ_k can be parsed as $(\mathsf{dbsp}_k, \mathsf{com}_k, \pi_k = (\pi_{\mathsf{GS},k}, \pi_{\mathsf{lin},k}, \pi_{\mathsf{trunc},k}))$ for $k \in \{0,1\}$. Without loss of generality, we assume that $\mathcal{A}$ makes the RO queries to be made during verification. Now, we consider the following sequence of games (which are analogous to the games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$ in the unforgeability proof).

Game $\mathbf{G}_0^{\mathcal{A}}(\lambda)$. This game is exactly the linkability game.

Game $\mathbf{G}_1^{\mathcal{A}}(\lambda)$. This game generates the CRS for GS, $\mathsf{crs}_{\mathsf{GS}}$ along with a trapdoor td using the extractor set up $\mathsf{Ext}_{\mathsf{Setup}}$ as defined in F-knowledge of GS. It is easy to see that there exists an adversary $\mathcal{B}_{\mathsf{dist}}$ playing the CRS indisintguishability game of $\mathsf{Ext}_{\mathsf{Setup}}$ such that

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

Game $\mathbf{G}_2^{\mathcal{A}}(\lambda)$. In this game, the game uses the extractor $\mathsf{Ext}_{\mathsf{P}}(\mathsf{td}, \cdot)$ on

$$(\mathbb{x}_k = (\mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{pk}_{\mathsf{I}}, r_k, \mathsf{ctxt}_k, \mathsf{sn}_k, \mathsf{dbsp}_k, \mathsf{com}_k), \pi_{\mathsf{GS},k})$$

for each $k \in [QN + 1]$ to get the witness $\widetilde{\mathbb{w}}^{(k)}$ of the relation $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ containing the following group elements (omitting the superscript $(\cdot)^{(k)}$ for readability) that satisfies Equation (1).

$$\begin{aligned}
&\widehat{\mathsf{cnt}}_1 \in \mathbb{G}_1, \widehat{\mathsf{cnt}}_2 \in \mathbb{G}_2,\\
&\mathsf{pk}_{\mathsf{User}} \in \mathbb{G}_1, C_\gamma \in \mathbb{G}_1,\\
&C = (C_{f,1}, C_{f,2}, (C_{g,i}, C_{T,1,i}, C_{T,2,i})_{i\in[3]}) \in \mathbb{G}_1^{11}, \sigma,\\
&(Y_j \in \mathbb{G}_1, \overline{Y}_{1,j} \in \mathbb{G}_1, \overline{Y}_{2,j} \in \mathbb{G}_2, \mathsf{open}_{f,j} = (B'_{f,j}, Q_{f,j}) \in \mathbb{G}_1^2)_{j\in[2]},\\
&(Z_i \in \mathbb{G}_1, \mathsf{open}_{g,i} = (B'_{g,i}, Q_{g,i}) \in \mathbb{G}_1^2, (\hat{T}_{j,i} \in \mathbb{G}_1, \mathsf{open}_{T,j,i} \in \mathbb{G}_1)_{j\in[2]})_{i\in[3]},\\
&\boldsymbol{\Gamma}_0, \boldsymbol{\Gamma}_1 \in \mathbb{G}_2^2
\end{aligned}$$

The game aborts if $(\mathbb{x}_k, \widetilde{\mathbb{w}}^{(k)}) \notin \widetilde{\mathcal{R}}_{\mathsf{tok}}$. By perfect F-knowledge of GS,

$$\mathsf{Pr}[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] = \mathsf{Pr}[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1] .$$

Now, we also consider several cases where $\mathcal{A}$ can win in game $\mathbf{G}_2$.

- $\mathsf{Forge}_{\mathsf{SPS}}$: This is defined as in the unforgeability proof, i.e., it is the event where one of the extracted commitments $(\mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$ was not signed in any issuance oracle queries. Note that by the same analysis as in the unforgeability proof. There exists an adversary $\mathcal{B}_{\mathsf{SPS}}$ such that $\mathsf{Pr}[\mathsf{Forge}_{\mathsf{SPS}}] \leqslant \mathsf{Adv}_{\mathsf{SPS}}^{\mathsf{unf}}(\mathcal{B}_{\mathsf{SPS}}, \lambda)$ making at most Q_{Iss} queries to its signing oracle.
- $\mathsf{BadForm}$: There exists some $k \in \{0, 1\}$ such that $\mathbb{w}^{(k)}$ contains $\widehat{\mathsf{cnt}}_1^{(k)}, (Y_j^{(k)}, \overline{Y}_{1,j}^{(k)})_{j\in[2]}$ such that one of the following is true: (a) $\mathrm{dlog}_{G_1} \widehat{\mathsf{cnt}}_1^{(k)} \notin [0, N - 1]$, (b) $\mathrm{dlog}_{G_1} Y_j^{(k)} \neq \mathrm{dlog}_{G_1} \overline{Y}_{1,j}^{(k)} \pmod{2^m}$, or (c) $\mathrm{dlog}_{G_1} \overline{Y}_{1,j}^{(k)} \notin [0, 2^m - 1]$. Similar to the unforgeability proof, there exists an adversary $\mathcal{B}_{\Pi}$, guessing which of the two tokens contain the group elements with the mentioned format and output the corresponding proofs, such that $\mathsf{Pr}[\mathsf{BadForm}] \leqslant 2\mathsf{Adv}_{\Pi_{\mathsf{trunc}}}^{\mathsf{sound}}(\mathcal{B}_{\Pi}, \lambda)$.
- Coll_γ: For two issuance oracle calls, the issuer's message imsg for each of them contain the same C_γ. In this case, since $C_\gamma = \gamma_{0,1}H_1 + \gamma_{0,2}H_2 + \gamma_{1,1}H_3 + \gamma_{1,2}H_4$, the probability that this event occurs is at most Q_{Iss}^2/p.
- BadG: Let i_0, i_1 be the index to the issuance query corresponding to $(\mathsf{pk}_{\mathsf{User}}^{(0)}, C^{(0)}, C_\gamma^{(0)})$ and $(\mathsf{pk}_{\mathsf{User}}^{(0)}, C^{(1)}, C_\gamma^{(1)})$, respectively. The event BadG correspond to the event where the underlying $\boldsymbol{\gamma}_{0,i_b}, \boldsymbol{\gamma}_{1,i_b}$ that the game samples during issuance are distinct from the discrete logarithms of $\boldsymbol{\Gamma}_0^{(b)}, \boldsymbol{\Gamma}_1^{(b)}$ extracted via $\mathsf{Ext}_{\mathsf{P},\mathsf{GS}}$.
- When $\mathsf{Forge}_{\mathsf{SPS}}$, $\mathsf{BadForm}$, Coll_γ, BadG do not occur (in this case, from $\widehat{\mathsf{cnt}}_1^{(k)}$ and $\overline{Y}_{1,j}^{(k)}$ the game can efficiently compute $\mathsf{cnt}^{(k)}, \overline{y}_j^{(k)}$ since they lie in a small range), we have the one of the following events:
  - $\mathsf{BadLink}$: The following extracted values are identical for $k = 0, 1$

    $$(\widehat{\mathsf{cnt}}_1^{(k)}, \mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)}) .$$

    However, since $\mathcal{A}$ wins the game, the output of $\mathsf{Identify}$ does not output $\mathsf{pk}' \in \mathcal{U}$.
  - SNColl: The extracted tuple $(\mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$ are identical for $k = 0, 1$, but $\widehat{\mathsf{cnt}}_1^{(0)} \neq \widehat{\mathsf{cnt}}_1^{(1)}$. In particular, the values are associated with the same issuance oracle query.
  - SNColl_2: The extracted tuple $(\mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$ are distinct for $k = 0, 1$ (it might be the case that $\widehat{\mathsf{cnt}}_1^{(0)} = \widehat{\mathsf{cnt}}_1^{(1)}$). In particular, the values are associated with two different issuance oracle queries.

Note that $\mathsf{Pr}[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] \leqslant Q_{\mathrm{Iss}}^2/p + \mathsf{Pr}[\mathsf{Forge}_{\mathsf{SPS}}] + \mathsf{Pr}[\mathsf{BadForm}] + \mathsf{Pr}[\mathsf{BadLink}] + \mathsf{Pr}[\mathsf{BadG}] + \mathsf{Pr}[\mathsf{SNColl}] + \mathsf{Pr}[\mathsf{SNColl}_2]$. We now analyze each case separately. In particular, the following lemmas, proved in Sections C.1 to C.4, capture the probability bounds on $\mathsf{Pr}[\mathsf{BadLink}], \mathsf{Pr}[\mathsf{BadG}], \mathsf{Pr}[\mathsf{SNColl}], \mathsf{Pr}[\mathsf{SNColl}_2]$. Then, the bound in the theorem follows.

Lemma 6.5. *There exists adversaries* $\mathcal{B}_{\mathsf{ebind}}, \mathcal{B}_{\mathsf{pbind}}, \mathcal{B}_{\mathrm{dlog},1}$ *all running in time roughly* $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_1(\lambda)} \ln(\frac{16}{\varepsilon_1(\lambda)}) t_{\mathcal{A}}$ *where* $\varepsilon_1(\lambda) \geqslant \mathsf{Pr}[\mathsf{BadLink} \vee \mathsf{BadForm}]$ *such that*

$$\mathsf{Pr}[\mathsf{BadLink}] \leqslant 8(\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{ebind}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},1}, \lambda)) + 7\mathsf{Pr}[\mathsf{BadForm}] \ .$$

Lemma 6.6. *There exists an adversary* $\mathcal{B}_{\mathrm{dlog},2}$ *running in time roughly* $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_2(\lambda)} \ln(\frac{16}{\varepsilon_2(\lambda)}) t_{\mathcal{A}}$ *where* $\varepsilon_2(\lambda) \geqslant \mathsf{Pr}[\mathsf{BadG}]$ *such that*

$$\mathsf{Pr}[\mathsf{BadG}] \leqslant 16\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},2}, \lambda) \ .$$

Lemma 6.7. *There exists an adversary* $\mathcal{B}_{\mathrm{dlog},3}$ *running in time* $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_3(\lambda)} \ln(\frac{16}{\varepsilon_3(\lambda)}) t_{\mathcal{A}}$ *where* $\varepsilon_3(\lambda) = Q_{\mathrm{Iss}}^{-1} \mathsf{Pr}[\mathsf{SNColl} \vee \mathsf{BadForm}]$ *and adversaries* $\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}$ *and* $\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}$ *running in time* $\frac{256k\lambda Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_3 \varepsilon_{\mathsf{SNColl}}} \ln(\frac{16}{\varepsilon_3}) t_{\mathcal{A}}$ *where* $\varepsilon_{\mathsf{SNColl}}(\lambda) \geqslant \frac{\varepsilon_3(\lambda)}{8} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda) - \mathsf{Pr}[\mathsf{BadForm}]$ *such that*

$$\begin{aligned}\mathsf{Pr}[\mathsf{SNColl}] \leqslant 8Q_{\mathrm{Iss}}(&\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}, \lambda) + 2^{-2^{m+1}\lambda+2} + \\ &+ \mathsf{Pr}[\mathsf{BadForm}] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda)) \ .\end{aligned}$$

Lemma 6.8. *There exists an adversary* $\mathcal{B}_{\mathrm{dlog},4}$ *running in time* $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_4(\lambda)} \ln(\frac{16}{\varepsilon_4(\lambda)}) t_{\mathcal{A}}$ *where* $\varepsilon_4(\lambda) = Q_{\mathrm{Iss}}^{-2} \mathsf{Pr}[\mathsf{SNColl}_2 \vee \mathsf{BadForm}]$ *and adversaries* $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}$ *and* $\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}$ *running in time* $\frac{256k\lambda Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_4 \varepsilon_{\mathsf{SNColl}_2}} \ln(\frac{16}{\varepsilon_4}) t_{\mathcal{A}}$ *where* $\varepsilon_{\mathsf{SNColl}_2}(\lambda) \geqslant \frac{\varepsilon_4(\lambda)}{8} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},4}, \lambda) - \mathsf{Pr}[\mathsf{BadForm}]$ *such that*

$$\begin{aligned}\mathsf{Pr}[\mathsf{SNColl}_2] \leqslant 8Q^2_{\mathrm{Iss}}(&\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}, \lambda) + 2^{-2^{m+1}\lambda+2} + \\ &+ \mathsf{Pr}[\mathsf{BadForm}] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},4}, \lambda)) \ .\end{aligned}$$

Finally, we can conclude the bound in the theorem as

$$\begin{aligned}\mathsf{Adv}^{\mathsf{link}}_{\mathsf{EARLT},k,N}(\mathcal{A}, \lambda) \leqslant\ &\mathsf{Adv}^{\mathsf{dist}}_{\mathsf{Ext}_{\mathsf{Setup}}}(\mathcal{B}_{\mathsf{dist}}, \lambda) + \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SPS}}(\mathcal{B}_{\mathsf{SPS}}, \lambda) + \frac{Q^2_{\mathrm{Iss}}}{p} + \frac{Q^2_{\mathrm{Iss}}}{2^{2\lambda}} \\ &+ 16(Q^2_{\mathrm{Iss}} + Q_{\mathrm{Iss}} + 1)\mathsf{Adv}^{\mathsf{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{B}_{\Pi}, \lambda) + 16\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},2}, \lambda) \\ &+ 8(\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{ebind}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},1}, \lambda)) \\ &+ 8Q_{\mathrm{Iss}}(\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda)) \\ &+ 8Q^2_{\mathrm{Iss}}(\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},4}, \lambda)) \ .\end{aligned}$$

□

6.4 Exculpability

Theorem 6.9 (Exculpability of EARLT). *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order* $p = p(\lambda)$*,* EARLT = EARLT[GGen]*, and* $k = k(\lambda), N = N(\lambda)$ *be integers. For any adversary* $\mathcal{A}$ *playing the* EXCULP *game of* EARLT *with running time* $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$*, there exists an adversary* $\mathcal{B}_{\mathsf{anon}}$ *(playing* ANON *game of* EARLT *with the simulator* Sim *defined as in Theorem 6.1) and* $\mathcal{B}_{\mathrm{dlog}}$ *running in time* $\approx t_{\mathcal{A}}$ *such that*

$$\mathsf{Adv}^{\mathsf{exculp}}_{\mathsf{EARLT},k,N}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{\mathsf{anon}}_{\mathsf{EARLT},k,N,\mathsf{Sim}}(\mathcal{B}_{\mathsf{anon}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}}, \lambda) \ .$$

Also, $\mathcal{B}_{\mathsf{anon}}$ *makes the same amount of queries to its oracles as that of* $\mathcal{A}$*.*

Proof. We first consider the following sequence of games.
Game $\mathbf{G}_0^{\mathcal{A}}(\lambda)$. This game is identical to the $\text{EXCULP}_{\mathsf{EARLT},k,N}$ game.
Game $\mathbf{G}_1^{\mathcal{A}}(\lambda)$. The game now samples crs using the anonymity simulator and simulates the honest users' interaction using the simulator as well. In particular, this game does the following:

Setup: Run the $(\mathsf{crs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda)$.
Init and new user oracles: These are run as in the exculpability game, i.e., registering the issuer's public key and sampling new users' secret and public keys using UKGen.
User oracle: When the adversary query $\mathrm{U}_1, \mathrm{U}_2$ oracle, the game runs $\mathsf{Sim}_{\mathsf{User}}$ as in the anonymity game.
Show oracle: On input (ctxt, R), run and return $(\mathsf{sn}, \tau) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{Show}}(\mathsf{td}, \mathsf{pk}_\mathsf{I}, \mathsf{ctxt}, R)$.

By anonymity of EARLT, we have that there exists an adversary $\mathcal{B}_{\mathsf{anon}}$ playing the anonymity game of EARLT such that

$$\Pr[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1] \geqslant \Pr[\mathbf{G}_0^{\mathcal{A}}(\lambda) = 1] - \mathsf{Adv}^{\mathsf{anon}}_{\mathsf{EARLT},k,N}(\mathcal{B}_{\mathsf{anon}}, \lambda) \ .$$

More precisely, this reduction takes advantage of the (almost) identical format of exculpability and anonymity games and use the oracles from anonymity game to simulate the oracles in the exculpability game.

Game $\mathbf{G}_2$. In this game, the honest user generation oracle is now replied with random group elements $\mathsf{pk}_{\mathsf{User}} \leftarrow\!\!\$\ \mathbb{G}_1$ without the game knowing the discrete logarithm of $\mathsf{pk}_{\mathsf{User}}$.

$$\Pr[\mathbf{G}_2^{\mathcal{A}}(\lambda) = 1] = \Pr[\mathbf{G}_1^{\mathcal{A}}(\lambda) = 1] \ .$$

Note that in game $\mathbf{G}_2$, if the adversary wins the game, the identification algorithm outputs the public key of an honest user. Also, by how the identification algorithm is defined, the game also learns the secret key. Hence, we can defined a reduction $\mathcal{B}_{\text{dlog}}$ playing the DL game such that on input $((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}), G_1, X_1)$, it replies to the i-th honest user generation oracle queries with $\mathsf{pk}_i \leftarrow \alpha_i G_1 + \beta_i X_1$ with $\alpha_i \leftarrow\!\!\$\ \mathbb{Z}_p, \beta_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$. At the end of the game, when $\mathcal{A}$ wins in game $\mathbf{G}_1$, it outputs two tokens $(\mathsf{sn}_0^*, \tau_0^*), (\mathsf{sn}*_1, \tau_1^*)$ on the same context ctxt but different nonces $r_0 \neq r_1$ such that $\mathsf{sn}_1^* = \mathsf{sn}_0^*$ and $\mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_\mathsf{I}, R_0, R_1, \mathsf{sn}_0^*, \tau_0^*, \tau_1^*) = \mathsf{pk}_{i*}$ for some honest user public key pk_{i*} tied to the i^*-th query to NewUsr.

Importantly, with how our $\mathsf{Identify}$ algorithm is defined, $\mathcal{B}_{\text{dlog}}$ would derive sk^* such that $\mathsf{sk}^* G_1 = \mathsf{pk}_{i*}$ along the way. Hence, $\mathcal{B}_{\text{dlog}}$ can derive and return $x' = (\mathsf{sk}^* - \alpha_{i*}) \cdot \beta_{i*}^{-1}$. Since the view of $\mathcal{A}$ is identical to its view in $\mathbf{G}_2$ (as each pk_i are uniformly random in $\mathbb{G}_1$), we have that

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

Acknowledgements

Anna Lysyanskaya was supported by NSF Grants 2312241, 2154170, and 2247305 as well as the Ethereum Foundation. Rutchathon Chairattana-Apirom and Stefano 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. Nico Döttling is funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

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.

AFG+10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Berlin, Heidelberg, August 2010.

AFK22. Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 113–142. Springer, Cham, November 2022.

AGJ+24. Sven Argo, Tim Güneysu, Corentin Jeudy, Georg Land, Adeline Roux-Langlois, and Olivier Sanders. Practical post-quantum signatures for privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 1523–1537. ACM Press, October 2024.

AHN+23. Masayuki Abe, Dennis Hofheinz, Ryo Nishimaki, Miyako Ohkubo, and Jiaxin Pan. Compact structure-preserving signatures with almost tight security. *Journal of Cryptology*, 36(4):37, October 2023.

AJO+19. Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, Jiaxin Pan, Arnab Roy, and Yuyu Wang. Shorter QA-NIZK and SPS with tighter security. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 669–699. Springer, Cham, December 2019.

AJOR18. Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy. Improved (almost) tightly-secure simulation-sound QA-NIZK with applications. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 627–656. Springer, Cham, December 2018.

ARF24. The european digital identity wallet architecture and reference framework, 2024. Accessed: 2025-02-13.

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.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BCC04. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004.

BCJ08. Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008.

BGY25. Foteini Baldimtsi, Rishab Goyal, and Aayush Yadav. Batched & non-interactive blind signatures from lattices. Cryptology ePrint Archive, Paper 2025/1771, 2025.

BHKN19. Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness-preserving reductions via cuckoo hashing. *Journal of Cryptology*, 32(2):361–392, April 2019.

BIP+18. Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 699–729. Springer, Cham, November 2018.

Bow17. Sean Bowe. BLS12-381: New zk-snark elliptic curve construction. https://electriccoin.co/blog/new-snark-curve/, 2017.

CDL16. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *Trust and Trustworthy Computing: 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings 9*, pages 1–20. Springer, 2016.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Berlin, Heidelberg, August 1994.

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.

CHL05. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Berlin, Heidelberg, May 2005.

CHL06. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 141–155. Springer, Berlin, Heidelberg, September 2006.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Berlin, Heidelberg, May 2001.

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.

Dam00. Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430. Springer, Berlin, Heidelberg, May 2000.

DGS+18. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.

DY05. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Berlin, Heidelberg, January 2005.

EG14. Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Berlin, Heidelberg, March 2014.

FK23. Dankrad Feist and Dmitry Khovratovich. Fast amortized KZG proofs. Cryptology ePrint Archive, Report 2023/033, 2023.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

GHKP18. Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 230–258. Springer, Cham, April / May 2018.

GHO20. Sanjam Garg, Mohammad Hajiabadi, and Rafail Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 OT and more. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 88–116. Springer, Cham, November 2020.

Gro04. Jens Groth. *Honest verifier zero-knowledge arguments applied*. PhD thesis, Citeseer, 2004.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008.

JR17. Charanjit S. Jutla and Arnab Roy. Improved structure preserving signatures under standard bilinear assumptions. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 183–209. Springer, Berlin, Heidelberg, March 2017.

JS25. Corentin Jeudy and Olivier Sanders. Improved lattice blind signatures from recycled entropy. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 477–513, Cham, 2025. Springer Nature Switzerland.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.

LM19. Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Cham, August 2019.

LPS24. Helger Lipmaa, Roberto Parisella, and Janno Siim. Constant-size zk-SNARKs in ROM from falsifiable assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 34–64. Springer, Cham, May 2024.

PP08. Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM Journal on Computing*, 38(1):85–96, 2008.

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.

YW25. Cathie Yun and Christopher A. Wood. Anonymous Rate-Limited Credentials. Internet-Draft draft-yun-cfrg-arc-01, Internet Engineering Task Force, August 2025. Work in Progress.

A Security of Pagh-Pagh Function Family

Since our instantiation of Pagh-Pagh function family does not choose f_1, f_2 as d-wise independent function, but only almost d-wise independent, we will give a prove of the security anew. Note that most of the proof will follow the results of [PP08, BHKN19]. We note again that the functions $f_1, f_2 : \mathbb{Z}_p \to [ck]$ for some integer $c \geqslant 8$ are chosen by (1) uniformly sampling $\mathbb{Z}_p$-polynomials q_1, q_2 of degree d, and (2) when evaluating the function f_1 on input $x \in \mathbb{Z}_p$, compute $q_1(x) \pmod{ck}$ (for some constant $c > 1$ where ck is an integer). Note that f_1, f_2 are $\frac{ck}{p}$-almost d-wise independent, i.e., for any $x_1, \ldots, x_d \in \mathbb{Z}_p$ and $y_1, \ldots, y_d \in [0, ck-1]$,

$$\mathsf{Pr}_f[\forall i \in [d], f(x_i) = y_i] \leqslant \left(\frac{1}{ck} + \frac{1}{p}\right)^d .$$

To see this, for $y \in [0, ck-1]$, let $S_y := \{y' \in [0, p-1] : y' \pmod{ck} = y\}$ which has size at most $\lfloor p/ck \rfloor + 1$. Then, we can write $\mathsf{Pr}_f[\forall i \in [d], f(x_i) = y_i] = \mathsf{Pr}_{q \leftarrow\$ \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]}[\forall i \in [d], q(x_i) \in S_{y_i}]$. Since q is of degree at most d, this probability is $\prod_{i\in[d]} \mathsf{Pr}_{q \leftarrow\$ \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]}[q(x_i) \in S_{y_i}] \leqslant \left(\frac{\lfloor p/ck \rfloor + 1}{p}\right)^d$.

Now, we restate a lemma from [BHKN19] which states the condition for a function family $\mathcal{F}$ to be adaptively secure (pseudorandom to be more precise) against any bounded query adversary.

Definition A.1 ([BHKN19, Def. 3.1]). *Let $\mathcal{S}, \mathcal{T}$ be sets. A set $\mathcal{M} \subseteq \mathcal{S}^* \times \mathcal{T}$[7] is "left-monotone" if for every $(\bar{s}_1, t) \in \mathcal{M}$ and every $\bar{s}_2 \in \mathcal{S}^*$ that has $\bar{s}_1$ as a prefix, it holds that $(\bar{s}_2, t) \in \mathcal{M}$.*

Lemma A.2 ([BHKN19, Lemma 3.2]). *Let $\mathcal{U}, \mathcal{V}$ be non-empty sets, let $\mathcal{F} = \mathcal{F}(\mathcal{U}, \mathcal{V}) = \{f_{u,v} : \mathcal{D} \to \mathcal{R}\}_{(u,v)\in\mathcal{U}\times\mathcal{V}}$ be a function family and let $\mathsf{Bad} \subseteq \mathcal{D}^* \times \mathcal{U}$ be left-monotone. Let $k \in \mathbb{N}$ and assume that for every $\bar{q} = (q_1, \ldots, q_{|\bar{q}|}) \in \mathcal{D}^{\leqslant k}$ it holds that:*

1. *$(f(q_1), \ldots, f(q_{|\bar{q}|}))_{f \leftarrow\$ \{f_{u,v} : v\in\mathcal{V}\}}$ is uniform over $\mathcal{R}^{|\bar{q}|}$ for every $u \in \mathcal{U}$ with $(\bar{q}, u) \notin \mathsf{Bad}$, and*
2. *$\mathsf{Pr}_{u \leftarrow\$ \mathcal{U}}[(\bar{q}, u) \in \mathsf{Bad}] \leqslant \varepsilon$.*

Then, for any (unbounded) adversary $\mathcal{A}$ making at most k query to its oracle and Π being the set of all functions from $\mathcal{D}$ to $\mathcal{R}$.

$$|\mathsf{Pr}_{u \leftarrow\$ \mathcal{U}, v \leftarrow\$ \mathcal{V}}[\mathcal{A}^{f_{u,v}} = 1] - \mathsf{Pr}_{\pi \leftarrow\$ \Pi}[\mathcal{A}^{\pi} = 1]| \leqslant \varepsilon$$

We will now show the following lemma establishing the properties for $\mathcal{F}$ to apply Lemma A.2.

Lemma A.3. *Let $k, d \in \mathbb{N}$ with $k/2 \geqslant d \geqslant 32$, $c \geqslant 8$ be a constant, and $p > 216k$ be a prime. Let $\mathcal{U} = (\mathbb{Z}_p^{\leqslant d}[\mathsf{X}])^2$ and $\mathcal{V} = (\mathbb{Z}_p^{ck} \times \mathbb{Z}_p^{ck} \times \mathbb{Z}_p^{\leqslant d}[\mathsf{X}])^\ell$. The function family $\mathcal{F}(\mathcal{U}, \mathcal{V})$ containing functions*

$$F_{f_1,f_2,(T_{1,i},T_{2,i},g_i)_{i\in[\ell]}} : x \mapsto (T_{1,i}[f_1(x) \pmod{ck}] + T_{2,i}[f_2(x) \pmod{ck}] + g_i(x))_{i\in[\ell]} .$$

Then, there exists a left-monotone $\mathsf{Bad} \subseteq (\mathbb{Z}_p)^* \times \mathcal{U}$ such that for any $\bar{q} = (q_1, \ldots, q_{|\bar{q}|}) \in \mathbb{Z}_p^{\leqslant k}$

(1) $(F_{f_1,f_2,v}(q_1), \ldots, F_{f_1,f_2,v}(q_{|\bar{q}|}))_{v \leftarrow\$ \mathcal{V}}$ is uniformly random in $\mathbb{Z}_p^\ell$ for any $(f_1, f_2) \in \mathcal{U}$ such that $(\bar{q}, (f_1, f_2)) \notin \mathsf{Bad}$.

(2) $\mathsf{Pr}_{(f_1,f_2)\leftarrow\mathcal{U}}[(\bar{q}, (f_1, f_2)) \in \mathsf{Bad}] \leqslant \frac{k}{2^{d/2-6}}$.

Proof. The proof follows that of [PP08]. First, define for any polynomial $f_1, f_2 \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$ and $\bar{q} \subseteq \mathbb{Z}_p$ a bipartite graph $G(f_1, f_2, \bar{q}) = (A, B, E)$ where $A = \{a_1, \ldots, a_{ck}\}, B = \{b_1, \ldots, b_{ck}\}$ are vertex sets and the edge set $E = \{e_x = (a_{f_1(x) \pmod{ck}}, b_{f_2(x) \pmod{ck}}) : x \in \bar{q}\}$. We say that a subgraph $E' \subseteq E$ is leafless if there is no vertex with *exactly one* incident edge. Accordingly, we define Bad as a set of $(\bar{q}, (f_1, f_2))$ such that $G(f_1, f_2, \bar{q})$ has a leafless subgraph E' with $|E'| > d$.

[7] $\mathcal{S}^*$ denotes the power set of $\mathcal{S}$.

We refer to the full proof of [PP08, Lemma 3.3] for (1). At a high-level, for $x \in \bar{q}$ such that the edge e_x incident to a leaf, $F(x)$ will be uniformly random because $T_{1,i}, T_{2,i}$ are uniformly random. Then, one can inductively peel off the edges incident to a leaf (by arguing that these are uniformly random), leaving the leafless subgraph of size at most d. Finally, if the size of leafless subgraph E' of $G(f_1, f_2, \bar{q})$ is at most d, because $(g_i)_{i\in[\ell]}$ are random $\mathbb{Z}_p$-polynomials with degree at most d, we have that $F(x)$ are uniformly random for all x such that $e_x \in E'$.

To show (2), following the proof in [PP08], we will bound the probability (over f_1, f_2) that the leafless subgraph E' of $G(f_1, f_2, \bar{q})$ is such that $|E'| \geqslant d$. Assume w.l.o.g. that d is even and $d \leqslant k$. Then, when $|E'| \geqslant d$, either there is a connected leafless subgraph with size at least $d/2$ or we have a leafless subgraph of size d' with $d/2 < d' \leqslant d$. These properties imply that there exists a subgraph of d' edges and at most $d'+1$ vertices where $d/2 \leqslant d' \leqslant d$. Hence, we will bound the probability that $G(f_1, f_2, \bar{q})$ contains such subgraph.

We now count the number of different edge labeled subgraphs with such property which we do so by bounding the following:

- Number of ways to choose d' edge labels out of k edges: bounded by $\binom{k}{d'} \leqslant (ek/d')^{d'}$.
- Number of ways to choose at most $d'+1$ vertices out of $2ck$ vertices: bounded by $\sum_{i=1}^{d'+1} \binom{2ck}{i} \leqslant (2ck/(d'+1))^{d'+1}$ (Note: $d \leqslant k/2$.)
- Let d_a, d_b be number of vertices in A, B chosen by the above (where $d_a + d_b \leqslant d'+1$). Then, the number of vertex assignments are at most $(d_a d_b)^{d'} \leqslant \left(\frac{d_a+d_b}{2}\right)^{2d'} \leqslant ((d'+1)/2)^{2d'}$. (The first inequality follows from AM-GM).

From the above, we have that the number of subgraph with d' edges and up to $d'+1$ vertices are at most

$$\left(\frac{ek}{d'}\right)^{d'} \left(\frac{2eck}{d'}\right)^{d'+1} \left(\frac{d'+1}{2}\right)^{2d'} = \frac{2eck}{d'+1} \left(\frac{ce^2k^2}{2} \cdot \frac{d'+1}{d'}\right)^{d'} \leqslant \frac{2eck}{d'+1} \left(4ck^2\right)^{d'} .$$

The last inequality follows from $d' \geqslant d/2 \geqslant 16$.

Now, we bound the probability that each of such subgraphs is sampled which is at most $\left(\frac{1}{ck} + \frac{1}{p}\right)^{2d'}$ by how f_1, f_2 are defined. Hence, summing over all values of d', we have the bound of

$$\begin{aligned}
&\sum_{d'=d/2}^{d} \frac{2eck}{d'+1} (4ck^2)^{d'} \left(\frac{1}{ck} + \frac{1}{p}\right)^{2d'} \\
&\leqslant \sum_{d'=d/2}^{d} \frac{2eck}{d'+1} \left(4ck^2((ck)^{-2} + 3(ckp)^{-1})\right)^{d'} \\
&\leqslant \sum_{d'=d/2}^{d} \frac{2eck}{d'+1} \left(\frac{4}{c} + 12\frac{k}{p}\right)^{d'} \\
&\leqslant (d/2+1) \frac{2eck}{d/2+1} \left(\frac{4}{c} + 12\frac{k}{p}\right)^{d/2}
\end{aligned}$$

For $c \geqslant 8$, we have that

$$\mathsf{Pr}_{(f_1,f_2)\leftarrow\mathcal{U}}[(\bar{q}, (f_1, f_2)) \in \mathsf{Bad}] \leqslant \frac{16ek}{2^{d/2}} + \frac{192ek^2d^2}{2^{d/2}p} \leqslant \frac{k}{2^{d/2-6}} . \square$$

Proof (of Lemma 4.4.). The lemma follows from Lemmas A.2 and A.3, setting $\ell = 3$ and $c = 8$. $\square$

B Security of KZG Commitments

For the sake of completeness, we include additional security proofs for properties of KZG polynomial commitment schemes and the vector commitment scheme derived from KZG in Sections B.1 and B.2.

B.1 Degree-Binding of $\mathsf{KZG}_{\mathsf{Ped}}$

Recall again that for KZG commitments and $d = d(\lambda)$, the degree-binding advantage of any adversary $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG},d}(\mathcal{A},\lambda) :=$

$$\mathsf{Pr}\left[\begin{array}{l|l}(\forall i \in [d+2] : \mathsf{KZG.V}(\mathsf{crs}, C, \alpha_i, \beta_i, \pi_i) = 1) \wedge & (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda) \\ (\forall f \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}], \exists i \in [d+2] : \quad f(\alpha_i) \neq \beta_i) & \mathsf{crs} \leftarrow\!\!\$\ \mathsf{KZG.Setup}((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}), d) \\ & (C, (\alpha_i, \beta_i, \pi_i)_{i\in[d+2]}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs})\end{array}\right] .$$

Note that the winning condition can be checked efficiently by computing f, g (via Lagrange interpolation) that passes through all (α_i, β_i) and (α_i, β_i') and check the degree if it is at most d.

Lemma B.1. *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order* $p = p(\lambda)$ *and* $d = d(\lambda)$*. Then, for any adversary* $\mathcal{A}$ *running in time* $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$*, there exists an adversary* $\mathcal{B}$ *running in time roughly that of* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG},d}(\mathcal{A},\lambda) \leqslant \mathsf{Adv}^{d\text{-}\mathsf{arsdh}}_{\mathsf{GGen}}(\mathcal{B},\lambda) .$$

Proof. Consider when $\mathcal{A}$ wins in its game, i.e., outputting $(C, (\alpha_i, \beta_i, \pi_i)_{i\in[d+2]})$ that satisfies the winning condition. Then, we have that either there exists valid openings $(\alpha_i, \beta_i, \pi_i), (\alpha_j, \beta_j, \pi_j)$ such that $\alpha_i = \alpha_j$ and $\beta_i \neq \beta_j$, or α_i are all distinct. For the former, we have that this breaks evaluation-binding of KZG, which is implied by ARSDH assumption. For the latter, we have that the lowest degree of $f \in \mathbb{Z}_p[\mathsf{X}]$ such that $f(\alpha_i) = \beta_i$ for all $i \in [d+2]$ is $d+1$. Now, define f_j as the lowest possible degree polynomial that $f_i(\alpha_j) = \beta_j$ for all $j \in [d+2]\backslash\{i\}$. Note that because $\deg f > d, f_j \neq f_i$ for some $i \neq j$. Fix those i, j. Now, we consider two cases:

- $C = f_i(x)G_1 = f_j(x)G_1$ (which can be checked given $xG_1, \ldots, x^dG_1$). Then, since $f_i \neq f_j$, we have that $f_i(x) - f_j(x) = 0$. Then, x can be computed by factoring the polynomial $f_i(\mathsf{X}) - f_j(\mathsf{X})$.
- $C \neq f_i(x)G_1$ or $C \neq f_j(x)G_1$. Let f^* be such that $f^*(x)G_1 \neq C$. In this case, we have $d+1$ openings $(\alpha_k^*, \beta_k^*, \pi_k^*)_{k\in[d+1]}$ such that $f^*(\alpha_k^*) = \beta_k^*$. Using the reduction from [LPS24, Theorem 1.], given x^iG_1 for $i = 1, \ldots, d$, one can compute $S = \{\alpha_k^*\}_{k\in[d+1]}$, group elements $A = C - f^*(x)G_1$ and $B = \sum_{k=1}^{d+1}(Z_{S\backslash\{\alpha_k^*\}}(\alpha_k^*))\pi_k^*$, where $Z_{S'}(\mathsf{X}) = \prod_{s\in S'}(\mathsf{X} - s)$ for any $S' \subseteq \mathbb{Z}_p$. Then, by [LPS24, Lemma 2.], $A = Z_S(x)B$, and this breaks the d-ARSDH assumption.

In both cases, we break ARSDH assumption. □

For the perfectly hiding KZG commitment $\mathsf{KZG}_{\mathsf{Ped}}$, we have the following lemma

Lemma B.2. *Let* GGen *be a bilinear group parameters generator outputting groups of prime-order* $p = p(\lambda)$ *and* $d = d(\lambda)$*. Then, for any adversary* $\mathcal{A}$ *running in time* $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$*, there exists an adversary* $\mathcal{B}, \mathcal{B}'$*, and* $\mathcal{B}''$ *running in time roughly that of* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{A},\lambda) \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B},\lambda) + \mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG},d}(\mathcal{B}',\lambda) + \mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}'',\lambda) .$$

Proof. First, recall that when $\mathcal{A}$ wins in its game, it returns $(C, (\alpha_i, \beta_i, \pi_i)_{i\in[d+2]})$ that satisfies the winning condition. Moreover, $\pi_i = (\beta_i', \pi_i')$ such that

$$\mathsf{e}\left(C - \beta_i G_1 - \beta_i' H, G_2\right) = \mathsf{e}\left(\pi_i', X_{2,1} - \alpha_i G_2\right) .$$

Consider the case that some $\alpha_i = \alpha_j$ and $\beta_i \neq \beta_j$. Then, this breaks evaluation-binding of $\mathsf{KZG}_{\mathsf{Ped}}$ commitment. Otherwise, all α_i's are distinct. Now, let $\alpha = \mathrm{dlog}_{G_1}(H)$ and $f, g \in \mathbb{Z}_p^{\leqslant d+1}[\mathsf{X}]$ be such that $f(\alpha_i) = \beta_i, g(\alpha_i) = \beta_i'$ for all $i \in [d+2]$. Since $\mathcal{A}$ wins in the game, $\deg f > d$. Now, we consider two cases:

- $\deg(f(\mathsf{X}) + \alpha g(\mathsf{X})) \leqslant d$: Since $\deg f > d$, $\deg g > d$ as well. Therefore, with $\deg(f + \alpha g) \leqslant d$, we have that the coefficients $f_{d+1}, g_{d+1} \neq 0$ of X^{d+1} for both f and g are such that $f_{d+1} + \alpha g_{d+1} = 0$. Thus, we can extract the discrete log α.

- $\deg(f(\mathsf{X}) + \alpha g(\mathsf{X})) > d$: In this case, we simply reduce to the degree-binding property of plain KZG commitment. In particular, the reduction $\mathcal{B}'$ receives the KZG CRS $(\mathsf{par}, (X_{1,i})_{i\in[0,d]}, G_2, X_{2,1})$, sample $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$, and run $\mathcal{A}$ on input $(\mathsf{par}, G_1, H = \alpha G_1, (X_{1,i}, \alpha X_{1,i})_{i\in[d]}, G_2, X_{2,1})$. On the output $(C, (\alpha_i, \beta_i, (\beta'_i, \pi'_i))_{i\in[d+2]})$ of $\mathcal{A}$, return $(C, (\alpha_i, \beta_i + \alpha\beta'_i, \pi'_i)_{i\in[d+2]})$. The advantage of $\mathcal{B}'$ is then easy to argue.

The proof concludes by applying the union bound. □

The bound in Lemma 4.2 follows from Lemmas B.1 and B.2. □

B.2 Security of $\mathsf{VC}_{\mathsf{KZG}}$

We now give the proof for Lemma 4.3. For **correctness**, this follows from correctness of KZG from Lemma 4.1.

To argue **statistical hiding**, we first consider when the discrete logarithm x of the group elements $X_{1,1}$ in the CRS is not in $[0, S-1]$. Then, we have that the following Vandermonde matrix

$$V = \begin{pmatrix} 1 \ldots & 0^S \\ \vdots \ddots & \vdots \\ 1 \ldots & (S-1)^S \\ 1 \ldots & x^S \end{pmatrix} \in \mathbb{Z}_p^{(S+1)\times(S+1)}$$

is full-rank. Now, we consider any vector $\boldsymbol{v} \in \mathbb{Z}_p^S$. We will show that the probability that the commitment of $\boldsymbol{v}$ sampled from $\mathsf{VC}_{\mathsf{KZG}}$ is some group element $C \in \mathbb{G}_1$ is $1/p$. To see this, notice that the polynomial f committed by $\mathsf{KZG.Com}$ is a uniformly random polynomial of degree $\leqslant S$ with the condition that $f(i) = v_i$ for all $i \in [0, S-1]$. Then, the desired probability is

$$\begin{aligned}\mathsf{Pr}_{f \leftarrow\!\!\$\ \mathbb{Z}_p^{\leqslant S}[\mathsf{X}]}[f(x)G_1 = C | \forall i \in [0,S-1], f(i) = v_i] &= \frac{\mathsf{Pr}_{f \leftarrow\!\!\$\ \mathbb{Z}_p^{\leqslant S}[\mathsf{X}]}[f(x)G_1 = C \ \wedge\ \forall i \in [0,S-1], f(i) = v_i]}{\mathsf{Pr}_{f \leftarrow\!\!\$\ \mathbb{Z}_p^{\leqslant S}[\mathsf{X}]}[\forall i \in [0,S-1], f(i) = v_i]} \\ &= p^{-S-1}/p^{-S} = \frac{1}{p}\end{aligned}$$

The second equality follows from V being full-rank. Thus, the advantage of any $\mathcal{A}_{\mathsf{hide}}$ will be bounded by the probability that $x \in [0, S-1]$ which is at most S/p.

For **position-binding**, observe that any adversary $\mathcal{A}_{\mathsf{pbind}}$ outputting two different openings $v_i \neq v'_i$ to the same position i will immediately break evaluation-binding of KZG. The bound then follows from Lemma 4.1.

Succinctness follows from the fact that each opening contains constant number of group elements and the verification perform constant number of group exponentiations and pairing evaluation. □

C Deferred Linkability Proofs

C.1 Proof of Lemma 6.5

We will use the same notations for variables as established in the proof of Theorem 6.4.

Recall that we want to bound the probability of the event $\mathsf{BadLink}$. In this case, we will employ the same proof strategy as with the event $\mathsf{AllSigned}$ in the unforgeability proof. In particular, we will define a wrapper $\mathcal{A}'$ (with the pseudocode in Figure 17) taking as input crs, sk_{I}, pk_{I}, and a trapdoor td for the extractor of GS. (These are sampled according to the same distribution as in $\mathbf{G}_2$.) The adversary $\mathcal{A}'$ has access to the random oracle $\mathsf{H}_{\mathsf{lin}}$ making at most Q_{H} queries. It runs the adversary $\mathcal{A}$ as in $\mathbf{G}_2$ (simulating the issuance with sk_{I}, the $\mathsf{H}_{\mathsf{lin}}$ queries are replied by querying its own RO, and $\mathsf{H}_{\mathsf{trunc}}$ queries are simulated using its own random coins). At the end of the game, checks whether $(\widehat{\mathsf{cnt}}_1^{(k)}, \mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$ are identical for $k \in \{0,1\}$ and neither $\mathsf{Forge}_{\mathsf{SPS}}$ nor Coll_γ are invoked (this can be done efficiently as $\mathcal{A}'$ knows the trapdoor).

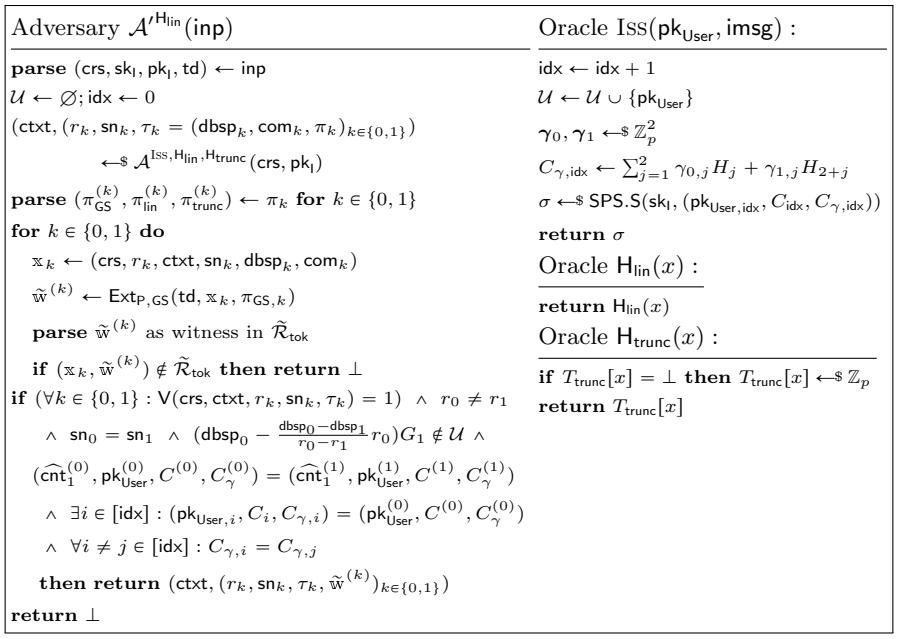

Fig. 17. Wrapper adversary $\mathcal{A}'$ for proof of Lemma 6.5

[8] If BadLink occurs, it returns $(\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau = (\mathsf{dbsp}_k, \mathsf{com}_k, \pi_k), \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}})$. With how $\mathcal{A}'$ is defined $\mathsf{Pr}[\mathcal{A}'(\mathsf{inp}) \neq \perp] = \mathsf{Pr}[\mathsf{BadLink} \vee \mathsf{BadForm}]$.

Similar to the unforgeability proof, we want to apply the extractor $\mathsf{Ext}_{\mathsf{lin}}$ from Lemma 5.1 with $L = 2$ on $\mathcal{A}'$ as follows. First, run the wrapper $\mathcal{A}'$ on input inp with the random coins $\rho_{\mathcal{A}'}$ and simulate the oracle $\mathsf{H}_{\mathsf{lin}}$ to $\mathcal{A}'$ (letting $\boldsymbol{h}$ be the RO outputs). Then, on the output $\mathsf{out} = (\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k, \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}})$ of $\mathcal{A}'$, run the extractor $\mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}'}(\mathsf{inp}, \mathsf{out}, \boldsymbol{h}; \rho_{\mathcal{A}'})$ which returns the witnesses $\mathbb{w}_{\mathsf{lin}}^{(0)}, \mathbb{w}_{\mathsf{lin}}^{(1)}$, containing the scalar openings.

Denote the event Good_1 as the event that (1) the extraction succeeds, (2) BadLink occurs with respect to the output of $\mathcal{A}'$, and (3) the extracted witness $\mathbb{w}_{\mathsf{lin}}^{(k)}$ defined as

$$\begin{aligned}\mathbb{w}_{\mathsf{lin}}^{(k)} = ((y_j^{(k)}, \mathsf{rand}_{y_j}^{(k)}, {\beta'_{f,j}}^{(k)}, \mathsf{rand}_{\beta'_{f,j}}^{(k)})_{j\in[2]},\\ (z_i^{(k)}, \mathsf{rand}_{z_i}^{(k)}, {\beta'_{g,i}}^{(k)}, \mathsf{rand}_{\beta'_{g,i}}^{(k)}, (t_{j,i}^{(k)}, \mathsf{rand}_{t_{j,i}}^{(k)})_{j\in[2]})_{i\in[3]}, \boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)}, \mathsf{rand}_{\boldsymbol{\gamma}_0}^{(k)}, \mathsf{rand}_{\boldsymbol{\gamma}_1}^{(k)})\end{aligned}$$

satisfy $y_j^{(k)} = \overline{y}_j^{(k)} \pmod{2^m}$ for $k \in \{0,1\}, j \in [2]$ (i.e., BadForm does not occur with respect to the outputs of $\mathcal{A}'$). Then, by Lemma 5.1, there exists a DL adversary $\mathcal{B}_{\mathrm{dlog},1}$ running in time $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_1(\lambda)} \ln(\frac{16}{\varepsilon_1(\lambda)})$ with $\varepsilon_1(\lambda) = \mathsf{Pr}[\mathsf{BadLink} \vee \mathsf{BadForm}]$, such that

$$\mathsf{Pr}[\mathsf{Good}_1 \vee \mathsf{BadForm}] \geqslant \frac{\mathsf{Pr}[\mathsf{BadLink} \vee \mathsf{BadForm}]}{8} - \mathsf{Adv}_{\mathsf{GGen}}^{\mathrm{dlog}}(\mathcal{B}_{\mathrm{dlog},1}, \lambda) .$$

Since Good_1 and BadForm are disjoint and similarly for BadLink and BadForm, we have that (using the fact that the distribution of $\mathsf{Ext}_{\mathsf{lin}}$ is the same as $\mathcal{A}'$)

$$\mathsf{Pr}[\mathsf{Good}_1] \geqslant \frac{\mathsf{Pr}[\mathsf{BadLink}] - 7\mathsf{Pr}[\mathsf{BadForm}]}{8} - \mathsf{Adv}_{\mathsf{GGen}}^{\mathrm{dlog}}(\mathcal{B}_{\mathrm{dlog},1}, \lambda) .$$

By the definition of $\mathcal{A}'$, if Good_1 occurs, we have the following:

[8] Note that we cannot check for BadForm, since it requires computing the discrete log $\mathrm{dlog}_{G_1} Y_j^{(k)}$ which is not guaranteed to be small.

- We can efficiently compute $\mathsf{cnt}^{(k)} = \mathrm{dlog}_{G_1} \widehat{\mathsf{cnt}}_1^{(k)}$ and $\overline{y}_j^{(k)} = \mathrm{dlog}_{G_1} \overline{Y}_j^{(k)}$ for $j \in [2]$ as $\mathsf{BadForm}$ does not occur.
- From the outputs of $\mathcal{A}'$, we have that $(\mathsf{cnt}^{(k)}, \mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$ are identical for $k \in \{0,1\}$.

We note that by how $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ is defined, we should have that

$$\mathsf{dbsp}^{(k)} = \mathrm{dlog}_{G_1} \mathsf{pk}_{\mathsf{User}}^{(k)} + (t_{1,3}^{(k)} + t_{2,3}^{(k)} + z_3^{(k)}) r_k \ .$$

Since $\mathsf{Identify}$ does not output $\mathsf{pk}_{\mathsf{User}}$ and $r_0 \neq r_1$ by the winning condition of $\mathbf{G}_2$, we have that $t_{1,3}^{(0)} + t_{2,3}^{(0)} + z_3^{(0)} \neq t_{1,3}^{(1)} + t_{2,3}^{(1)} + z_3^{(1)}$. Now, since we have that $\mathsf{cnt}^{(0)} = \mathsf{cnt}^{(1)}$, we can consider the following cases:

- $y_j^{(0)} \neq y_j^{(1)}$ for some $i \in [2]$. Here, we break evaluation-binding of $\mathsf{KZG}_{\mathsf{Ped}}$ commitment scheme $C_{f,j}^{(0)}$ with the openings $\mathsf{open}_{f,j}^{(k)} = (\beta'_{f,j}{}^{(k)}, Q_{f,j}^{(k)})$.
- $z_3^{(0)} \neq z_3^{(1)}$. Again, this also breaks evaluation-binding of $\mathsf{KZG}_{\mathsf{Ped}}$ commitment scheme on the commitment $C_{g,i}^{(0)}$.
- $y_j^{(0)} = y_j^{(1)}$ for all $i \in [2]$ and $t_{j,3}^{(0)} \neq t_{j,3}^{(1)}$ for some $j \in [2]$. Since $y_j^{(0)} = y_j^{(1)}$, we also have that $\overline{y}_j^{(0)} = \overline{y}_j^{(1)}$ (since $\mathsf{BadForm}$ does not occur), and accordingly with $t_{j,3}^{(0)} \neq t_{j,3}^{(1)}$, we break position-binding of $\mathsf{VC}_{\mathsf{KZG}}$ commitment $C_{T,j,3}$.

Therefore, there exists an adversary $\mathcal{B}_{\mathsf{ebind}}$ and $\mathcal{B}_{\mathsf{pbind}}$ playing the binding games of $\mathsf{KZG}_{\mathsf{Ped}}$ and $\mathsf{VC}_{\mathsf{KZG}}$ commitment schemes, respectively such that

$$\Pr[\mathsf{Good}_1] \leqslant \mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{ebind}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}}, \lambda) \ .$$

Hence,

$$\begin{aligned}\Pr[\mathsf{BadLink}] &\leqslant 8(\Pr[\mathsf{Good}_1] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},1}, \lambda)) + 7\Pr[\mathsf{BadForm}] \\ &\leqslant 8(\mathsf{Adv}^{\mathsf{ebind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{ebind}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{pbind}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},1}, \lambda)) \\ &\quad + 7\Pr[\mathsf{BadForm}] \ . \square\end{aligned}$$

C.2 Proof of Lemma 6.6

Recall that we want to bound the probability of the event BadG. In this case, we will define a wrapper $\mathcal{A}'$ (with the pseudocode in Figure 18) taking as input crs, sk_{I}, pk_{I}, and a trapdoor td for the extractor of GS. (These are sampled according to the same distribution as in $\mathbf{G}_2$.) The adversary $\mathcal{A}'$ has access to the random oracle $\mathsf{H}_{\mathsf{lin}}$ making at most Q_{H} queries. It runs the adversary $\mathcal{A}$ as in $\mathbf{G}_2$ (simulating the issuance with sk_{I}, the $\mathsf{H}_{\mathsf{lin}}$ queries are replied by querying its own RO, and $\mathsf{H}_{\mathsf{trunc}}$ queries are simulated using its own random coins). At the end of the game, checks whether BadG is invoked (this can be done efficiently as $\mathcal{A}'$ knows the trapdoor so it can extract $\boldsymbol{\Gamma}_0^{(k)}, \boldsymbol{\Gamma}_1^{(k)}$). If BadG occurs, it returns $(\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k, \widetilde{\mathbb{W}}^{(k)}, \boldsymbol{\gamma}_{0,i_k}, \boldsymbol{\gamma}_{1,i_k})_{k\in\{0,1\}})$ where i_0, i_1 are defined as in the event BadG. With how $\mathcal{A}'$ is defined $\Pr[\mathcal{A}'(\mathsf{inp}) \neq \bot] = \Pr[\mathsf{BadG}]$.

Then, we want to apply the extractor $\mathsf{Ext}_{\mathsf{lin}}$ from Lemma 5.1 with $L = 2$ on $\mathcal{A}'$ as follows. First, run the wrapper $\mathcal{A}'$ on input inp with the random coins $\rho_{\mathcal{A}'}$ and simulate the oracle $\mathsf{H}_{\mathsf{lin}}$ to $\mathcal{A}'$ (letting $\boldsymbol{h}$ be the RO outputs). Then, on the output $\mathsf{out} = (\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k, \widetilde{\mathbb{W}}^{(k)})_{k\in\{0,1\}})$ of $\mathcal{A}'$, run the extractor $\mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}'}(\mathsf{inp}, \mathsf{out}, \boldsymbol{h}; \rho_{\mathcal{A}'})$ which returns the witnesses $\mathbb{w}_{\mathsf{lin}}^{(0)}, \mathbb{w}_{\mathsf{lin}}^{(1)}$. Denote the event Good_2 as the event that the extraction succeeds. Also denote the extracted witnesses as

$$\begin{aligned}\mathbb{w}_{\mathsf{lin}}^{(k)} = ((&y_j^{(k)}, \mathsf{rand}_{y_j}^{(k)}, \beta'_{f,j}{}^{(k)}, \mathsf{rand}_{\beta'_{f,j}}^{(k)})_{j\in[2]}, \\ &(z_i^{(k)}, \mathsf{rand}_{z_i}^{(k)}, \beta'_{g,i}{}^{(k)}, \mathsf{rand}_{\beta'_{g,i}}^{(k)}, (t_{j,i}^{(k)}, \mathsf{rand}_{t_{j,i}}^{(k)})_{j\in[2]})_{i\in[3]}, \boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)}, \mathsf{rand}_{\gamma_0}^{(k)}, \mathsf{rand}_{\gamma_1}^{(k)}) \ .\end{aligned}$$

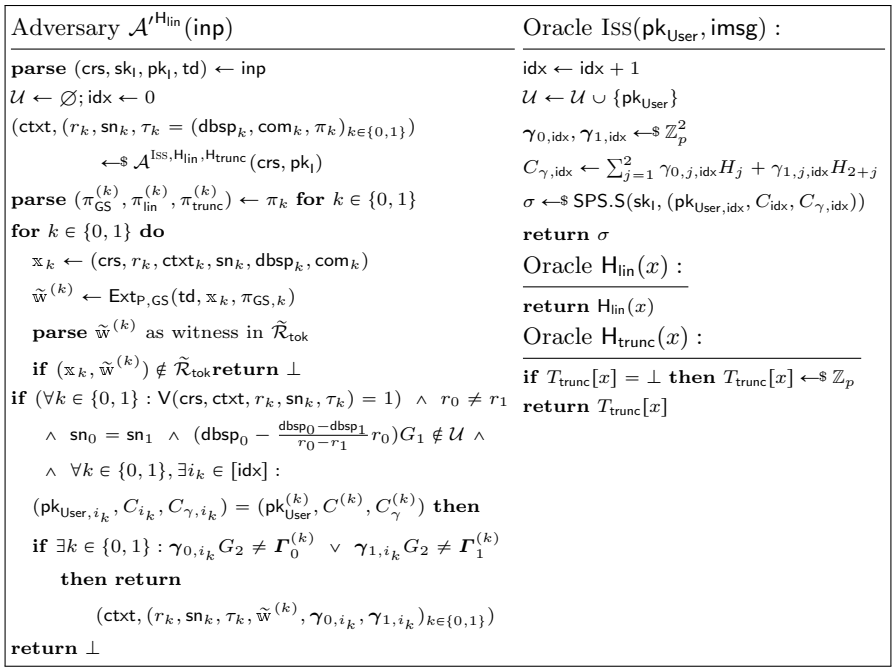

Fig. 18. Wrapper adversary $\mathcal{A}'$ for proof of Lemma 6.6

Then, by Lemma 5.1, there exists a DL adversary $\mathcal{B}'_{\mathrm{dlog}}$ running in time $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_2(\lambda)} \ln(\frac{16}{\varepsilon_2(\lambda)}) t_{\mathcal{A}}$ with $\varepsilon_2(\lambda) = \mathsf{Pr}[\mathsf{BadG}]$, such that

$$\mathsf{Pr}[\mathsf{Good}_2] \geqslant \frac{\mathsf{Pr}[\mathsf{BadG}]}{8} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}'_{\mathrm{dlog}}, \lambda) \ .$$

Now, consider when Good_2 occurs. By how the event BadG is defined, the output $\boldsymbol{\gamma}_{0,i_k}, \boldsymbol{\gamma}_{1,i_k}$ contained in aux and $\boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)}$ extracted by $\mathsf{Ext}_{\mathsf{lin}}$ are such that for some $k \in \{0,1\}$,

$$(\boldsymbol{\gamma}_{0,i_k}, \boldsymbol{\gamma}_{1,i_k}) \neq (\boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)}) \ .$$

However, by how $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ is defined, with $H_1, \ldots, H_4$ contained in the crs (which is also the input of $\mathsf{Ext}_{\mathsf{lin}}$)

$$\sum_{j=1}^{2} \gamma_{0,j}^{(k)} H_j + \gamma_{1,j}^{(k)} H_{j+2} = C_{\gamma}^{(k)} = \sum_{j=1}^{2} \gamma_{0,i_k,j} H_j + \gamma_{1,i_k,j} H_{j+2} \ .$$

This leads to a non-trivial discrete logarithm relation over $H_1, \ldots, H_4$, breaking DL assumption. Hence, there exists an adversary $\mathcal{B}''_{\mathrm{dlog}}$ running in time roughly that of $\mathcal{B}'_{\mathrm{dlog}}$ such that $\mathsf{Pr}[\mathsf{Good}_2] \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}''_{\mathrm{dlog}}, \lambda)$. Thus, we can construct $\mathcal{B}_{\mathrm{dlog},2}$, by combining $\mathcal{B}'_{\mathrm{dlog}}$ and $\mathcal{B}''_{\mathrm{dlog}}$, such that

$$\mathsf{Pr}[\mathsf{BadG}] \leqslant 8(\mathsf{Pr}[\mathsf{Good}_2] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}'_{\mathrm{dlog}}, \lambda)) \leqslant 16\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},2}, \lambda) \ . \square$$

C.3 Proof of Lemma 6.7

Recall that we want to bound the probability of the event $\mathsf{BadLink}$. For this case, we will employ a rewinding argument to show that after enough rewinding, the adversary would break the position-binding or degree-binding property of the KZG commitment. Our proof strategy proceed in the following steps.

Wrapper $\mathcal{A}'$. We define a wrapper $\mathcal{A}'$ (given in Figure 19) taking as input crs, sk_{I}, pk_{I}, a trapdoor td for the extractor of GS, and vectors $\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^* \in \mathbb{Z}_p^2$. It then samples an index $i^* \leftarrow\!\$\ [Q_{\mathrm{Iss}}]$ (i.e., taking the

corresponding value in its random tape). Then, it runs $\mathcal{A}$ as in $\mathbf{G}_2$ with the exception that during the i^*-th issuance oracle call, it uses $\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*$ instead of drawing them from its random coins. At the end, when $\mathcal{A}$ returns its outputs, $\mathcal{A}'$ uses td to extract $\widetilde{\mathbb{w}}$ and $\widetilde{\mathbb{w}}'$, checks that (1) $\widehat{\mathsf{cnt}}_1^{(0)} \neq \widehat{\mathsf{cnt}}_1^{(1)}$ and the extracted $(\mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$ corresponds to the values signed during the i^*-th issuance oracle query for both $k \in \{0,1\}$, (2) $\mathcal{A}$ wins in the game $\mathbf{G}_2$, and (3) neither BadG nor Coll_γ occurs. (These checks can be made efficiently.) If these checks do not pass, $\mathcal{A}'$ aborts. Otherwise, it returns the outputs of $\mathcal{A}$ along with $\widetilde{\mathbb{w}}$ and $\widetilde{\mathbb{w}}'$. Hence, $\mathsf{Pr}[\mathcal{A}'(\mathsf{crs}, \mathsf{sk}_\mathsf{I}, \mathsf{pk}_\mathsf{I}, \mathsf{td}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*) \neq \perp] \geqslant \frac{1}{Q_{\mathrm{Iss}}} \mathsf{Pr}[\mathsf{SNColl} \vee \mathsf{BadForm}]$ (as $\mathcal{A}'$ guesses the index i^* and $\mathcal{A}'$ cannot rule out $\mathsf{BadForm}$ efficiently).

Wrapper $\mathcal{A}''$ rewinding $\mathcal{A}'$ to extract from π_{lin}. We define another wrapper $\mathcal{A}''$ (given in Figure 19) which takes as input crs, sk_I, pk_I, a trapdoor td for the extractor of GS, and vectors $\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^* \in \mathbb{Z}_p^2$. It runs the wrapper $\mathcal{A}'$ first on input inp with the random coins $\rho_{\mathcal{A}'}$ and $\mathcal{A}''$ simulates the oracle $\mathsf{H}_{\mathsf{lin}}$ to $\mathcal{A}'$ (letting $\boldsymbol{h}$ be the RO outputs). On the output $\mathsf{out} = (\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k, \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}})$ of $\mathcal{A}'$, $\mathcal{A}''$ runs the extractor $\mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}'}(\mathsf{inp}, \mathsf{out}, \boldsymbol{h}; \rho_{\mathcal{A}'})$ which returns the witnesses $\mathbb{w}_{\mathsf{lin}}^{(0)}, \mathbb{w}_{\mathsf{lin}}^{(1)}$. If the extractor aborts, $\mathcal{A}''$ aborts. Otherwise, the wrapper $\mathcal{A}''$ does the following:

- From $\widetilde{\mathbb{w}}^{(k)}$, compute the discrete logarithms of $\mathsf{cnt}^{(k)} = \mathsf{dlog}\widehat{\mathsf{cnt}}_1^{(k)}, \overline{y}_j^{(k)} = \mathsf{dlog}\overline{Y}_{1,j}^{(k)}$, which can be done efficiently if they are in the designated range $[0, N-1]$ and $[0, 2^m - 1]$.
- If the above cannot be extracted efficiently or $y_j^{(k)} \neq \overline{y}_j^{(k)} \pmod{2^m}$ ($y_j^{(k)}$ is defined in $\mathbb{w}_{\mathsf{lin}}^{(k)}$), abort. Note that this abort is equivalent to the event $\mathsf{BadForm}$ occurring in the output of $\mathcal{A}'$, i.e., with probability at most $\mathsf{Pr}[\mathsf{BadForm}]$.

If no abort, it returns ctxt and the following values for $k = 0, 1$ (these are parsed from $\widetilde{\mathbb{w}}^{(k)}, \mathbb{w}_{\mathsf{lin}}^{(k)}$)

$$\begin{aligned}&(\mathsf{sn}_k, C^{(k)}, \mathsf{cnt}^{(k)}, (y_j^{(k)}, \overline{y}_j^{(k)}, \mathsf{open}_{f,j}^{(k)})_{j\in[2]},\\ &\quad (t_{1,j}^{(k)}, t_{2,j}^{(k)}, z_j^{(k)}, \mathsf{open}_{T,1,j}^{(k)}, \mathsf{open}_{T,2,j}^{(k)}, \mathsf{open}_{g,j}^{(k)})_{j\in[2]}, \boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)}) \ .\end{aligned}$$

By how $\mathcal{A}'$ is defined, we have that for $k \in \{0,1\}$

$$\begin{aligned}\boldsymbol{\gamma}_0^* &= \boldsymbol{\gamma}_0^{(k)} \ ; \boldsymbol{\gamma}_1^* = \boldsymbol{\gamma}_1^{(k)}\\ \mathsf{sn}_k &= (t_{1,j}^{(k)} + t_{2,j}^{(k)} + z_j^{(k)})_{j\in[2]} + \mathsf{cnt}^{(k)} \cdot \boldsymbol{\gamma}_0^{(k)} + \boldsymbol{\gamma}_1^{(k)}\\ 1 &= \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{V}(\mathsf{crs}_{\mathsf{KZG}}, C_{f,j}^{(k)}, 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt} + \mathsf{cnt}^{(k)}, y_j^{(k)}, \mathsf{open}_{f,j}^{(k)}) \ , \forall j \in [2]\\ 1 &= \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{V}(\mathsf{crs}_{\mathsf{KZG}}, C_{g,i}^{(k)}, 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt} + \mathsf{cnt}^{(k)}, z_i^{(k)}, \mathsf{open}_{g,i}^{(k)}) \ , \forall i \in [2]\\ 1 &= \mathsf{VC}_{\mathsf{KZG}}.\mathsf{V}(\mathsf{crs}_{\mathsf{VC}}, C_{T,j,i}^{(k)}, \overline{y}_j^{(k)}, t_{j,i}^{(k)}, \mathsf{open}_{T,j,i}^{(k)}) \ , \forall j \in [2], i \in [2]\end{aligned}$$

Note also that by Lemma 5.1,

$$\begin{aligned}&\mathsf{Pr}[\mathcal{A}''(\mathsf{crs}, \mathsf{sk}_\mathsf{I}, \mathsf{pk}_\mathsf{I}, \mathsf{td}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*) \neq \perp]\\ &\geqslant \frac{\mathsf{Pr}[\mathsf{SNColl} \vee \mathsf{BadForm}]}{8Q_{\mathrm{Iss}}} - \mathsf{Adv}_{\mathsf{GGen}}^{\mathrm{dlog}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda) - \mathsf{Pr}[\mathsf{BadForm}]\\ &\geqslant \frac{\mathsf{Pr}[\mathsf{SNColl}]}{8Q_{\mathrm{Iss}}} - \mathsf{Adv}_{\mathsf{GGen}}^{\mathrm{dlog}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda) - \mathsf{Pr}[\mathsf{BadForm}] \ ,\end{aligned}$$

for some adversary $\mathcal{B}_{\mathrm{dlog},3}$ running in time $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_3}\ln(\frac{16}{\varepsilon_3})t_{\mathcal{A}}$ with $\varepsilon_3(\lambda) = \frac{1}{Q_{\mathrm{Iss}}}\mathsf{Pr}[\mathsf{SNColl} \vee \mathsf{BadForm}]$. Also, let $\varepsilon_{\mathsf{SNColl}}(\lambda) = \mathsf{Pr}[\mathcal{A}''(\mathsf{crs}, \mathsf{sk}_\mathsf{I}, \mathsf{pk}_\mathsf{I}, \mathsf{td}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*) \neq \perp]$. Note also that $\mathcal{A}''$ runs in time roughly $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_3}\ln(\frac{16}{\varepsilon_3})t_{\mathcal{A}}$ as well.

Rewind $\mathcal{A}''$ further to break degree-binding of KZG. We will construct an adversary $\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}$ and $\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}$ (see Figure 20) against degree-binding of $\mathsf{KZG}_{\mathsf{Ped}}$ and position-binding of $\mathsf{VC}_{\mathsf{KZG}}$, which rewind $\mathcal{A}''$ on different $\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^* \in \mathbb{Z}_p^2$ portion of the inputs for at most $K_{\max} = 2K/\varepsilon_{\mathsf{SNColl}}$ times until there are $K = 2^{m+3}\lambda$

successful runs. Note that all $\mathcal{A}''$ is run on the same inputs and random coins except for the $\boldsymbol{\gamma}_{0,l}, \boldsymbol{\gamma}_{1,l}$ which are given to the adversary $\mathcal{A}$ after it picks the commitments for the i^*-th issuance query. Hence, all outputs out_l that does not abort contains the same $(C^{(0)}, C^{(1)})$ commitments, and furthermore,

$$C^{(0)} = C^{(1)} = (C_{f,1}, C_{f,2}, (C_{g,i}, C_{T,1,i}, C_{T,2,i})_{i\in[3]})$$

as they correspond to the same issuance oracle query. Finally, the reduction tries to find the following:

- $\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}$: For some j, i, two valid openings $(\overline{y}, t, \mathsf{open}), (\overline{y}', t', \mathsf{open}')$ for $C_{T,j,i}$ with $\bar{y} = \bar{y}'$ and $t \neq t'$.
- $\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}$: Find
  - For some $j \in [2]$, $d+2$ valid openings for $C_{f,j}$, denoted as $S = \{(x_l = 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt}^{(l)} + \mathsf{cnt}^{(l)}, y_{j,l}, \mathsf{open}_{f,j,l})\}_{l\in[d+2]}$ such that the input-output pairs $(x_l, y_{j,l})$ do not lie on a degree $\leqslant d$ polynomial.
  - For some $i \in [2]$, $d+2$ valid openings for $C_{g,i}$, denoted as $S = \{(x_l = 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt}^{(l)} + \mathsf{cnt}^{(l)}, z_{i,l}, \mathsf{open}_{g,i,l})\}_{l\in[d+2]}$ such that the input-output pairs $(x_l, z_{i,l})$ do not lie on a degree $\leqslant d$ polynomial.

  These can be computed efficiently by incrementally constructing the set S and adding a new tuple $(x_l, y_l, \mathsf{open}_l)$ if (x_l, y_l) is not on the polynomial interpolating points in S. More precisely, we maintain a set S of tuples (x, y, open) and a polynomial $f \in \mathbb{Z}_p^{<|S|}[\mathsf{X}]$ such that $f(x) = y$ for all $(x, y, \mathsf{open}) \in S$. When we consider a new tuple $(x_l, y_l, \mathsf{open}_l)$, we check
  - If $f(x_l) = y_l$, then we do nothing and consider the next tuple.
  - Otherwise, we add $(x_l, y_l, \mathsf{open}_l)$ to S (call this new set S') and compute f' such that $f'(x) = y$ for all $(x, y, \mathsf{open}) \in S$. (This can be done via Lagrange interpolation.) If no such f' exists, this means there are two tuples with the same x but different y's; in this case, we just add tuples until the set is of size $d + 2$. Otherwise, we continue (with S', f') until we have a set of size $d + 2$.

If one of the above occurs, $\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}$ or $\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}$ wins in the game. Note that both run in time roughly $K_{\max} t_{\mathcal{A}''} = \frac{2^{m+4}\lambda t_{\mathcal{A}''}}{\varepsilon_{\mathsf{SNColl}}} \leqslant \frac{256k\lambda Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_3 \varepsilon_{\mathsf{SNColl}}} \ln(\frac{16}{\varepsilon_3}) t_{\mathcal{A}}$. (Note that $2^m = 8k$ is small enough as it is the size of the CRS.)

Analysis of $\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}, \mathcal{B}_{\mathsf{Coll},\mathsf{VC}}$. First, we let the event Bad be such that over the $K_{\max}$ runs of $\mathcal{A}''$, there exists a subset $L \subseteq [K_{\max}]$ of size K such that there exists $\boldsymbol{T}_{1,1}, \boldsymbol{T}_{1,2}, \boldsymbol{T}_{2,1}, \boldsymbol{T}_{2,2} \in \mathbb{Z}_p^{2^m}$ and polynomials $f_1, f_2, g_1, g_2 \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$ such that for each $l \in L$ there exists $\mathsf{ctxt}^{(l)} \in [0, 2^{\ell_{\mathsf{ctxt}}} - 1], \mathsf{cnt}^{(l,0)}, \mathsf{cnt}^{(l,1)} \in [0, N-1]$ where with $x^{(l,k)} = \mathsf{ctxt}^{(l)} 2^{\ell_{\mathsf{cnt}}} + \mathsf{cnt}^{(l,k)}$ for $k \in \{0, 1\}$

$$\begin{aligned}
&(\mathsf{cnt}^{(l,0)} - \mathsf{cnt}^{(l,1)}) \cdot \boldsymbol{\gamma}_{0,l} \\
&= (g_i(x^{(l,0)}) - g_i(x^{(l,1)}) + \sum_{j\in[2]} T_{i,j}[f_j(x^{(l,0)}) \mod 2^m] - T_{i,j}[f_j(x^{(l,1)}) \mod 2^m])_{i\in[2]} .
\end{aligned}$$

Note that the event Bad is defined over the choice of $\boldsymbol{\gamma}_{0,l}$ for $l \in [K_{\max}]$. Here, for a fixed $L \subseteq [K_{\max}]$ of size K, $\boldsymbol{T}_{1,1}, \boldsymbol{T}_{1,2}, \boldsymbol{T}_{2,1}, \boldsymbol{T}_{2,2} \in \mathbb{Z}_p^{2^m}$, polynomials $f_1, f_2, g_1, g_2 \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}]$, and $\mathsf{ctxt}^{(l)} \in [0, 2^{\ell_{\mathsf{ctxt}}} - 1], \mathsf{cnt}_0^{(l)}, \mathsf{cnt}_1^{(l)} \in [0, N-1]$, the probability over $\boldsymbol{\gamma}_{0,l}$ of the above equation being true is p^{-2K}.

Let Succ be the event that at least K of the $K_{\max}$ runs of $\mathcal{A}''$ does not abort. Note that if Bad does not occur, one of the winning condition of $\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}$ or $\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}$ will be satisfied. Hence,

$$\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{Coll},\mathsf{VC}}, \lambda) \geqslant \mathsf{Pr}[\mathsf{Succ}] - \mathsf{Pr}[\mathsf{Bad}] .$$

We can bound $\mathsf{Pr}[\mathsf{Bad}]$ via a union bound over all subsets $L \subseteq [K_{\max}]$ of size K and all possible vector and polynomials and $\mathsf{ctxt}^{(l)}, \mathsf{cnt}_0^{(l)}, \mathsf{cnt}_1^{(l)}$ for $l \in L$. Note that the probability is over the choices of $\boldsymbol{\gamma}_{0,l}$

$$\begin{aligned}
\mathsf{Pr}[\mathsf{Bad}] &\leqslant \binom{K_{\max}}{K} \cdot p^{4\cdot 2^m} \cdot p^{4d} \cdot (2^{\ell_{\mathsf{ctxt}}} N^2)^K \cdot p^{-2K} \\
&\leqslant \left(\frac{eK_{\max} pN}{Kp^{2-2^{m+3}/K}}\right)^K \leqslant \left(\frac{eK_{\max} N}{Kp^{1-2^{m+3}/K}}\right)^K
\end{aligned}$$

Adversary $\mathcal{A}'^{\mathsf{H}_{\mathsf{lin}}}(\mathsf{inp})$:

parse $(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*, \mathsf{td}) \leftarrow \mathsf{inp}$

$\mathcal{U} \leftarrow \varnothing$; $\mathsf{idx} \leftarrow 0$; $i^* \leftarrow\!\!\$\ [Q_{\mathrm{Iss}}]$

$(\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k = (\mathsf{dbsp}_k, \mathsf{com}_k, \pi_k)_{k\in\{0,1\}})$

$\quad \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{Iss},\mathsf{H}_{\mathsf{lin}},\mathsf{H}_{\mathsf{trunc}}}(\mathsf{crs}, \mathsf{pk}_{\mathsf{I}})$

parse $(\pi_{\mathsf{GS}}^{(k)}, \pi_{\mathsf{lin}}^{(k)}, \pi_{\mathsf{trunc}}^{(k)}) \leftarrow \pi_k$ **for** $k \in \{0,1\}$

for $k \in \{0,1\}$ **do**

$\quad \mathbb{x}_k \leftarrow (\mathsf{crs}, r_k, \mathsf{ctxt}_k, \mathsf{sn}_k, \mathsf{dbsp}_k, \mathsf{com}_k)$

$\quad \widetilde{\mathbb{w}}^{(k)} \leftarrow \mathsf{Ext}_{\mathsf{P},\mathsf{GS}}(\mathsf{td}, \mathbb{x}_k, \pi_{\mathsf{GS},k})$

$\quad$ **parse** $\widetilde{\mathbb{w}}^{(k)}$ as witness in $\widetilde{\mathcal{R}}_{\mathsf{tok}}$

$\quad$ **if** $(\mathbb{x}_k, \widetilde{\mathbb{w}}^{(k)}) \notin \widetilde{\mathcal{R}}_{\mathsf{tok}}$ **then return** $\bot$

if $(\forall k \in \{0,1\} : \mathsf{V}(\mathsf{crs}, \mathsf{ctxt}, r_k, \mathsf{sn}_k, \tau_k) = 1) \ \wedge\ r_0 \neq r_1$

$\quad \wedge\ \mathsf{sn}_0 = \mathsf{sn}_1 \ \wedge\ (\mathsf{dbsp}_0 - \frac{\mathsf{dbsp}_0 - \mathsf{dbsp}_1}{r_0 - r_1} r_0) G_1 \notin \mathcal{U} \ \wedge$

$\quad \forall k \in \{0,1\} : (\mathsf{pk}_{\mathsf{User},i^*}, C_{i^*}, C_{\gamma,i^*}) = (\mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})$

$\quad \wedge\ \widehat{\mathsf{cnt}}_1^{(0)} \neq \widehat{\mathsf{cnt}}_1^{(1)} \ \wedge\ \neg\mathsf{BadG} \ \wedge\ \neg\mathsf{Coll}_\gamma$ **then**

$\quad\quad$ **return** $(\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k, \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}})$

return $\bot$

Oracle $\mathrm{Iss}(\mathsf{pk}_{\mathsf{User}}, \mathsf{imsg})$:

$\mathsf{idx} \leftarrow \mathsf{idx} + 1$

if $\mathsf{idx} \neq i^*$ **then** $\boldsymbol{\gamma}_{0,\mathsf{cnt}}, \boldsymbol{\gamma}_{1,\mathsf{cnt}} \leftarrow\!\!\$\ \mathbb{Z}_p^2$

else $(\boldsymbol{\gamma}_{0,\mathsf{cnt}}, \boldsymbol{\gamma}_{1,\mathsf{cnt}}) \leftarrow (\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*)$

$C_{\gamma,\mathsf{idx}} \leftarrow \sum_{j=1}^2 \gamma_{0,j,\mathsf{idx}} H_j + \gamma_{1,j,\mathsf{idx}} H_{2+j}$

$\sigma \leftarrow\!\!\$\ \mathsf{SPS.S}(\mathsf{sk}_{\mathsf{I}}, (\mathsf{pk}_{\mathsf{User},\mathsf{idx}}, C_{\mathsf{idx}}, C_{\gamma,\mathsf{idx}}))$

return σ

Oracle $\mathsf{H}_{\mathsf{lin}}(x)$:

return $\mathsf{H}_{\mathsf{lin}}(x)$

Oracle $\mathsf{H}_{\mathsf{trunc}}(x)$:

if $T_{\mathsf{trunc}}[x] = \bot$ **then** $T_{\mathsf{trunc}}[x] \leftarrow\!\!\$\ \mathbb{Z}_p$

return $T_{\mathsf{trunc}}[x]$

Adversary $\mathcal{A}''(\mathsf{inp})$:

parse $(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*, \mathsf{td}) \leftarrow \mathsf{inp}$

$\rho_{\mathcal{A}'} \leftarrow\!\!\$\ \mathcal{R}_{\mathcal{A}'}$; $\mathsf{out} = (\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k = (\mathsf{dbsp}_k, \mathsf{com}_k, \pi_k, \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}}) \leftarrow\!\!\$\ \mathcal{A}'^{\mathsf{H}_{\mathsf{lin}}}(\mathsf{inp}; \rho_{\mathcal{A}'})$

// $\mathsf{H}_{\mathsf{lin}}$ is simulated via lazy sampling by $\mathcal{A}''$. Let $\boldsymbol{h}$ be the vector of RO outputs.

$((\mathrm{w}_{\mathsf{lin}}^{(k)})_{k\in\{0,1\}}) \leftarrow\!\!\$\ \mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}'}(\mathsf{inp}, \mathsf{out}, \boldsymbol{h}; \rho_{\mathcal{A}'})$

// For simplicity, assume $\mathsf{Ext}_{\mathsf{lin}}$'s input out is parsed in the format $((\mathbb{x}_k, \pi_{\mathsf{lin},k})_{k\in\{0,1\}}, \mathsf{aux})$.

if $(\mathrm{w}_{\mathsf{lin}}^{(k)})_{k\in\{0,1\}} = \bot$ **then return** $\bot$

parse $((y_j^{(k)}, \mathsf{rand}_{y_j}^{(k)}, {\beta'_{f,j}}^{(k)}, \mathsf{rand}_{\beta'_{f,j}}^{(k)})_{j\in[2]}, (z_i^{(k)}, \mathsf{rand}_{z_i}^{(k)}, {\beta'_{g,i}}^{(k)}, \mathsf{rand}_{\beta'_{g,i}}^{(k)},$

$\quad (t_{j,i}^{(k)}, \mathsf{rand}_{t_{j,i}}^{(k)})_{j\in[2]})_{i\in[3]}, \boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)}, \mathsf{rand}_{\boldsymbol{\gamma}_0}^{(k)}, \mathsf{rand}_{\boldsymbol{\gamma}_1}^{(k)}) \leftarrow \mathrm{w}_{\mathsf{lin}}^{(k)}$ **for** $k \in \{0,1\}$

parse $\widetilde{\mathbb{w}}^{(k)}$ as witness in $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ **for** $k \in \{0,1\}$

for $k \in \{0,1\}$ **do**

$\quad \mathsf{cnt}^{(k)} \leftarrow \mathrm{dlog}_{G_1} \widehat{\mathsf{cnt}}_1^{(k)}$ // If this takes too long, $\mathcal{A}''$ just aborts.

$\quad$ **for** $j \in [2]$: $\mathsf{open}_{f,j}^{(k)} \leftarrow ({\beta'_{f,j}}^{(k)}, Q_{f,j}^{(k)})$; $\overline{y}_j^{(k)} \leftarrow \mathrm{dlog}_{G_1} \overline{Y}_{1,j}^{(k)}$

$\quad$ **for** $i \in [2]$: $\mathsf{open}_{g,i}^{(k)} \leftarrow ({\beta'_{g,i}}^{(k)}, Q_{g,i}^{(k)})$

if $\exists k \in \{0,1\} : \mathsf{cnt}^{(k)} \notin [0, N-1] \ \vee\ (\exists j \in [2] : y_j^{(k)} \neq \overline{y}_j^{(k)} \pmod 2^m \ \vee\ \overline{y}_j^{(k)} \notin [0, 2^m - 1])$

$\quad$ **then return** $\bot$

return $(\mathsf{ctxt}, (C^{(k)}, \mathsf{cnt}^{(k)}, (y_j^{(k)}, \overline{y}_j^{(k)}, \mathsf{open}_{f,j}^{(k)})_{j\in[2]},$

$\quad (t_{1,j}^{(k)}, t_{2,j}^{(k)}, z_j^{(k)}, \mathsf{open}_{T,1,j}^{(k)}, \mathsf{open}_{T,2,j}^{(k)}, \mathsf{open}_{g,j}^{(k)})_{j\in[2]})_{k\in[2]})$

Fig. 19. Wrapper adversary $\mathcal{A}'$ and $\mathcal{A}''$ for proof of Lemma 6.7. Denote $\mathcal{R}_{\mathcal{A}'}$ as the randomness space of $\mathcal{A}'$.

The second inequality follows from $d \leqslant 2^m$, $\binom{n}{t} \leqslant (en/t)^t$ and $2^{\ell_{\mathsf{ctxt}}} N \leqslant p$. Setting $K = 2^{m+3}\lambda$ and $K_{\max} = 2K/\varepsilon_{\mathsf{SNColl}}$. Then, we have $\mathsf{Pr}[\mathsf{Bad}] \leqslant 2^{-2^{m+3}\lambda}$ with $\varepsilon_{\mathsf{SNColl}} \geqslant 8eN/p$.

We now analyze $\mathsf{Pr}[\mathsf{Succ}]$. Recall that $\varepsilon_{\mathsf{SNColl}} = \mathsf{Pr}[\mathcal{A}''$ does not abort$]$. Then, we first let G be a set of tuples $\mathsf{inp} = (\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \mathsf{td}, \rho_{\mathcal{A}''})$ such that

$$\mathsf{Pr}_{\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^* \leftarrow\!\!\$\ \mathbb{Z}_p^2}[\mathcal{A}''(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \mathsf{td}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*; \rho_{\mathcal{A}''}) \neq \bot] \geqslant \varepsilon_{\mathsf{SNColl}}/2 \ .$$

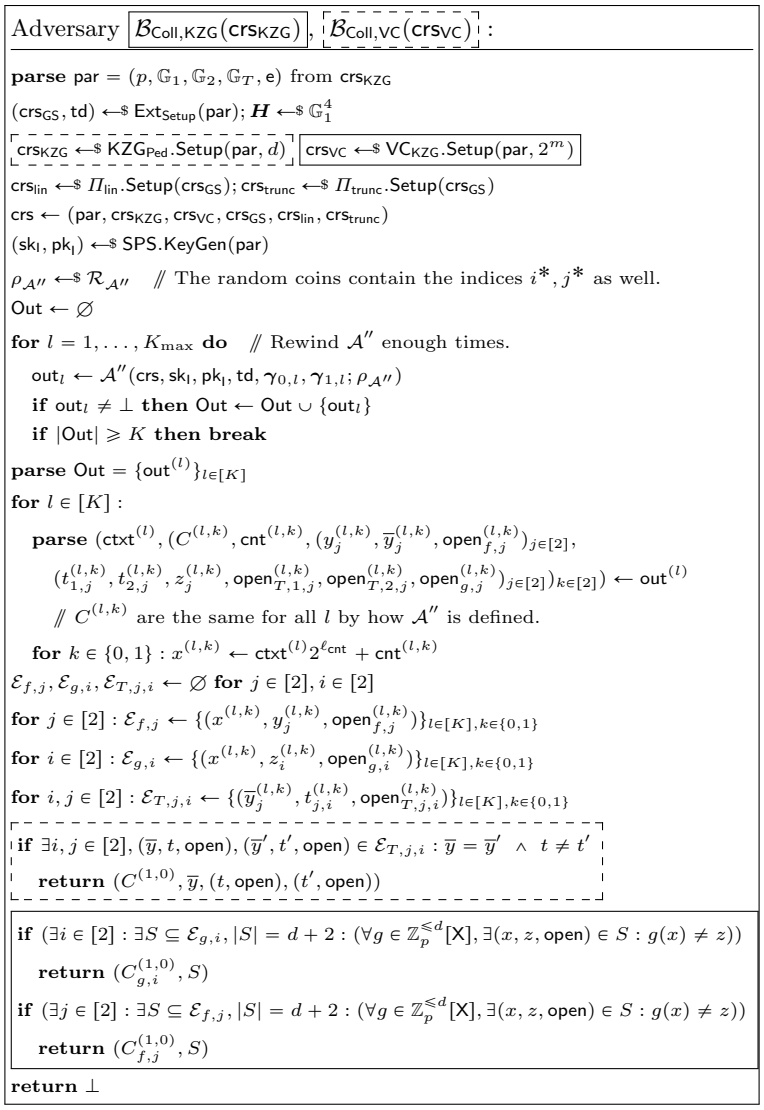

Fig. 20. Description of adversaries $\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}}, \mathcal{B}_{\mathsf{Coll},\mathsf{VC}}$. Denote $\mathcal{R}_{\mathcal{A}''}$ as the randomness space of $\mathcal{A}'$.

Then, one can see that $\mathsf{Pr}_{\mathsf{inp}}[\mathsf{inp} \in G] \geqslant \varepsilon_{\mathsf{SNColl}}/2$ as

$$\varepsilon_{\mathsf{SNColl}} \leqslant \mathsf{Pr}_{\mathsf{inp}}[\mathsf{inp} \in G] + (1 - \mathsf{Pr}_{\mathsf{inp}}[\mathsf{inp} \in G])\varepsilon_{\mathsf{SNColl}}/2$$
$$\varepsilon_{\mathsf{SNColl}}/2 \leqslant \mathsf{Pr}_{\mathsf{inp}}[\mathsf{inp} \in G](1 - \varepsilon_{\mathsf{SNColl}}/2) \leqslant \mathsf{Pr}_{\mathsf{inp}}[\mathsf{inp} \in G] \ .$$

Now, consider $\mathsf{Pr}[\mathsf{Succ}|\mathsf{inp} \in G]$, which corresponds to the probability of the event that at least K out of $K_{\max} = 2K/\varepsilon_{\mathsf{SNColl}}$ runs of $\mathcal{A}''$ does not abort. Let $X_1, \ldots, X_{K_{\max}}$ be indicators variable such that $X_i = 1$ if the i-th run of $\mathcal{A}''$ succeeds when $\mathsf{inp} \in G$. By definition of G, $\mathsf{Pr}[X_i = 1] \geqslant \varepsilon_{\mathsf{SNColl}}/2$. Also, let $X = \sum_i X_i$, $\mu = \mathsf{E}[X] \geqslant K_{\max}\varepsilon_{\mathsf{SNColl}} = 2K$. By Chernoff bound and with $K_{\max} = 2K/\varepsilon_{\mathsf{SNColl}}$ and $K = 2^{m+3}\lambda$, we have that

$$\mathsf{Pr}[\neg\mathsf{Succ}|\mathsf{inp} \in G] \leqslant \mathsf{Pr}[X \leqslant K] \leqslant \mathsf{Pr}[X \leqslant \frac{\mu}{2}] \leqslant e^{-\mu/8} \leqslant 2^{-2^{m+1}\lambda} \ .$$

Hence, $\mathsf{Pr}[\mathsf{Succ}] \geqslant (1 - 2^{-2^{m+1}\lambda})\varepsilon_{\mathsf{SNColl}}/2 \geqslant \varepsilon_{\mathsf{SNColl}}/2 - 2^{-2^{m+1}\lambda}$. Thus concluding the proof that

$$\mathsf{Pr}[\mathsf{SNColl}] \leqslant 8Q_{\mathrm{Iss}}(\mathsf{Pr}[\mathcal{A}'' \text{ does not abort}] + 2\mathsf{Adv}^{\mathsf{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{B}_{\mathsf{sound}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda)$$

$$\leqslant 8Q_{\mathrm{Iss}}(2\mathsf{Pr}[\mathsf{Succ}] + 2^{-2^{m+1}\lambda+1} + \mathsf{Pr}[\mathsf{BadForm}] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3},\lambda))$$
$$\leqslant 8Q_{\mathrm{Iss}}(\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}_{\mathsf{Coll},\mathsf{KZG}},\lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}_{\mathsf{Coll},\mathsf{VC}},\lambda) + 2^{-2^{m+1}\lambda+2} +$$
$$+ \mathsf{Pr}[\mathsf{BadForm}] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3},\lambda)) \ .\square$$

C.4 Proof of Lemma 6.8

We replicate the strategy for analyzing SNColl with the following specifications on the wrappers. The descriptions of the wrappers are given in Figure 21

- **Wrapper $\mathcal{A}'$.** The wrapper $\mathcal{A}'$ samples a pair of distinct indices $i^* < j^*$ (instead of only one i^* as in the prior proof) and uses $\boldsymbol{\gamma}^*_0, \boldsymbol{\gamma}^*_1$ in the j^*-th issuance oracle query. At the end, $\mathcal{A}'$ checks that the (1) the extracted $\{(\mathsf{pk}^{(k)}_{\mathsf{User}}, C^{(k)}, C^{(k)}_{\gamma})\}_{k\in\{0,1\}}$ corresponds to the values signed during the i^* and j^*-th issuance oracle queries, (2) $\mathcal{A}$ wins in the game $\mathbf{G}_2$, and (3) neither BadG nor Coll_γ occurs. (These checks can be made efficiently.) If these checks do not pass, $\mathcal{A}'$ abort. Otherwise, it returns the outputs of $\mathcal{A}$ along with $\widetilde{\mathbb{w}}$ and $\widetilde{\mathbb{w}}'$. Hence, $\mathsf{Pr}[\mathcal{A}'(\mathsf{crs},\mathsf{sk}_{\mathsf{I}},\mathsf{pk}_{\mathsf{I}},\mathsf{td},\boldsymbol{\gamma}^*_0,\boldsymbol{\gamma}^*_1) \neq \bot] \geqslant \frac{1}{Q^2_{\mathrm{Iss}}}\mathsf{Pr}[\mathsf{SNColl}_2 \ \vee \ \mathsf{BadForm}]$ (as $\mathcal{A}'$ guesses the index i^* and $\mathcal{A}'$ cannot rule out $\mathsf{BadForm}$ efficiently).
- **Wrapper $\mathcal{A}''$ rewinding $\mathcal{A}'$ to extract from π_{lin}.** Similar to the prior proof, runs the wrapper $\mathcal{A}'$ first on input inp with the random coins $\rho_{\mathcal{A}'}$ and $\mathcal{A}''$ simulates the oracle $\mathsf{H}_{\mathsf{lin}}$ to $\mathcal{A}'$ (letting $\boldsymbol{h}$ be the RO outputs). On the output $\mathsf{out} = (\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k, \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}})$ of $\mathcal{A}'$, $\mathcal{A}''$ runs the extractor $\mathsf{Ext}^{\mathcal{A}'}_{\mathsf{lin}}(\mathsf{inp},\mathsf{out},\boldsymbol{h};\rho_{\mathcal{A}'})$ which returns the witnesses $\mathbb{w}^{(0)}_{\mathsf{lin}}, \mathbb{w}^{(1)}_{\mathsf{lin}}$. If the extractor aborts, $\mathcal{A}''$ aborts. It also computes $\mathsf{cnt}^{(k)} = \mathrm{dlog}\widehat{\mathsf{cnt}}^{(k)}_1, \overline{y}^{(k)}_j = \mathrm{dlog}\overline{Y}^{(k)}_{1,j}$ and aborts if they are not of the right format (i.e., $\mathsf{BadForm}$ is invoked on the output of $\mathcal{A}'$). If no abort, it returns ctxt and the following values for $k = 0, 1$ (these are parsed from $\widetilde{\mathbb{w}}^{(k)}, \mathbb{w}^{(k)}_{\mathsf{lin}}$)

$$(\mathsf{sn}_k, C^{(k)}, \mathsf{cnt}^{(k)}, (y^{(k)}_j, \overline{y}^{(k)}_j, \mathsf{open}^{(k)}_{f,j})_{j\in[2]},$$
$$(t^{(k)}_{1,j}, t^{(k)}_{2,j}, z^{(k)}_j, \mathsf{open}^{(k)}_{T,1,j}, \mathsf{open}^{(k)}_{T,2,j}, \mathsf{open}^{(k)}_{g,j})_{j\in[2]}, \boldsymbol{\gamma}^{(k)}_0, \boldsymbol{\gamma}^{(k)}_1) \ .$$

By how $\mathcal{A}'$ is defined (i.e., Coll_γ and BadG does not occur and $\mathcal{A}$ wins in the game), we have that for $k \in \{0,1\}$

$$\mathsf{sn}_k = (t^{(k)}_{1,j} + t^{(k)}_{2,j} + z^{(k)}_j)_{j\in[2]} + \mathsf{cnt}^{(k)} \cdot \boldsymbol{\gamma}^{(k)}_0 + \boldsymbol{\gamma}^{(k)}_1$$
$$1 = \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{V}(\mathsf{crs}_{\mathsf{KZG}}, C^{(k)}_{f,j}, 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt} + \mathsf{cnt}^{(k)}, y^{(k)}_j, \mathsf{open}^{(k)}_{f,j}) \ , \forall j \in [2]$$
$$1 = \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{V}(\mathsf{crs}_{\mathsf{KZG}}, C^{(k)}_{g,i}, 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt} + \mathsf{cnt}^{(k)}, z^{(k)}_i, \mathsf{open}^{(k)}_{g,i}) \ , \forall i \in [2]$$
$$1 = \mathsf{VC}_{\mathsf{KZG}}.\mathsf{V}(\mathsf{crs}_{\mathsf{VC}}, C^{(k)}_{T,j,i}, \overline{y}^{(k)}_j, t^{(k)}_{j,i}, \mathsf{open}^{(k)}_{T,j,i}) \ , \forall j \in [2], i \in [2]$$

Note that $\boldsymbol{\gamma}^*_0 = \boldsymbol{\gamma}^{(k)}_0 ; \boldsymbol{\gamma}^*_1 = \boldsymbol{\gamma}^{(k)}_1$ for one of the $k \in \{0,1\}$ and $\boldsymbol{\gamma}^{(1-k)}_0, \boldsymbol{\gamma}^{(1-k)}_1$ will be identical to the value $\boldsymbol{\gamma}_{0,i^*}, \boldsymbol{\gamma}_{1,i^*}$ drawn from the random coins of $\mathcal{A}'$. Note also that by Lemma 5.1,

$$\mathsf{Pr}[\mathcal{A}''(\mathsf{crs},\mathsf{sk}_{\mathsf{I}},\mathsf{pk}_{\mathsf{I}},\mathsf{td},\boldsymbol{\gamma}^*_0,\boldsymbol{\gamma}^*_1) \neq \bot]$$
$$\geqslant \frac{\mathsf{Pr}[\mathsf{SNColl}_2 \ \vee \ \mathsf{BadForm}]}{8Q^2_{\mathrm{Iss}}} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},4},\lambda) - \mathsf{Pr}[\mathsf{BadForm}]$$
$$\geqslant \frac{\mathsf{Pr}[\mathsf{SNColl}_2]}{8Q^2_{\mathrm{Iss}}} - \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},4},\lambda) - \mathsf{Pr}[\mathsf{BadForm}] \ ,$$

for some adversary $\mathcal{B}_{\mathrm{dlog},4}$ running in time $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_4}\ln(\frac{16}{\varepsilon_4})t_{\mathcal{A}}$ with $\varepsilon_4(\lambda) = Q^{-2}_{\mathrm{Iss}}\mathsf{Pr}[\mathsf{SNColl}_2 \ \vee \ \mathsf{BadForm}]$. Let $\varepsilon_{\mathsf{SNColl}_2}(\lambda) = \mathsf{Pr}[\mathcal{A}''(\mathsf{crs},\mathsf{sk}_{\mathsf{I}},\mathsf{pk}_{\mathsf{I}},\mathsf{td},\boldsymbol{\gamma}^*_0,\boldsymbol{\gamma}^*_1) \neq \bot]$. Note that $\mathcal{A}''$ also runs in time $\frac{32Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_4}\ln(\frac{16}{\varepsilon_4})t_{\mathcal{A}}$.

Rewind $\mathcal{A}''$ to break degree-binding of KZG. We will construct an adversary $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}$ and $\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}$ (see Figure 22) against degree-binding of $\mathsf{KZG}_{\mathsf{Ped}}$ and position-binding of $\mathsf{VC}_{\mathsf{KZG}}$, which rewind $\mathcal{A}''$ on different $\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^* \in \mathbb{Z}_p^2$ portion of the inputs for at most $K_{\max} = 2K/\varepsilon_{\mathsf{SNColl}_2}$ times until there are $K = 2^{m+3}\lambda$ successful runs. Note that all $\mathcal{A}''$ is run on the same inputs and random coins except for the $\boldsymbol{\gamma}_{0,l}, \boldsymbol{\gamma}_{1,l}$ which are given to the adversary $\mathcal{A}$ after it picks the commitments for the j^*-th issuance query. Hence, all outputs out_l that does not abort contains the same $(C^{(0)}, C^{(1)})$ commitments. Without loss of generality, we also assume that $C^{(0)}$ corresponds to the commitment from the i^*-th issuance query. Since $i^* < j^*$, we also have that $\boldsymbol{\gamma}_0^{(0)}, \boldsymbol{\gamma}_1^{(0)}$ are the same overall outputs out_l. We also denote $C^{(k)}$ as

$$C^{(k)} = (C_{f,1}^{(k)}, C_{f,2}^{(k)}, (C_{g,i}^{(k)}, C_{T,1,i}^{(k)}, C_{T,2,i}^{(k)})_{i\in[3]}) \,.$$

Similar to the prior proof, $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}$ and $\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}$ will find the following: (the notations take from the reduction in Figure 22)

- $\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}$: For some $k \in \{0,1\}, j, i \in [2]$, two valid openings $(\overline{y}, t, \mathsf{open}), (\overline{y}', t', \mathsf{open}')$ for $C_{T,j,i}^{(k)}$ with $\overline{y} = \overline{y}'$ and $t \neq t'$.
- $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}$: Find
  - For some $k \in \{0,1\}, j \in [2]$, $d+2$ valid openings for $C_{f,j}^{(k)}$, denoted as $S = \{(x_l = 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt}^{(l)} + \mathsf{cnt}^{(l)}, y_{j,l}, \mathsf{open}_{f,j,l})\}_{l\in[d+2]}$ such that the input-output pairs $(x_l, y_{j,l})$ do not lie on a degree $\leqslant d$ polynomial.
  - For some $k \in \{0,1\}, i \in [2]$, $d+2$ valid openings for $C_{g,i}^{(k)}$, denoted as $S = \{(x_l = 2^{\ell_{\mathsf{cnt}}}\mathsf{ctxt}^{(l)} + \mathsf{cnt}^{(l)}, z_{i,l}, \mathsf{open}_{g,i,l})\}_{l\in[d+2]}$ such that the input-output pairs $(x_l, z_{i,l})$ do not lie on a degree $\leqslant d$ polynomial.

  These can be computed efficiently as in the proof of Lemma 6.7.

If one of the above exists, $\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}$ or $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}$ wins in the game. Note that both run in time roughly $K_{\max} t_{\mathcal{A}''} = \frac{2^{m+4}\lambda t_{\mathcal{A}''}}{\varepsilon_{\mathsf{SNColl}_2}} \leqslant \frac{256k\lambda Q_{\mathsf{H}_{\mathsf{lin}}}}{\varepsilon_4 \varepsilon_{\mathsf{SNColl}_2}} \ln(\frac{16}{\varepsilon_4}) t_{\mathcal{A}}$. (Note that $2^m = 8k$ is small enough as it is the size of the CRS.)

Analysis of $\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}$ and $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}$. The difference from the prior proof lies in how we define the event Bad. In particular, the event Bad is defined such that over the $K_{\max}$ runs of $\mathcal{A}''$, there exists a subset $L \subseteq [K_{\max}]$ of size K such that, there exists $\boldsymbol{T}_{1,1}^{(k)}, \boldsymbol{T}_{1,2}^{(k)}, \boldsymbol{T}_{2,1}^{(k)}, \boldsymbol{T}_{2,2}^{(k)} \in \mathbb{Z}_p^{2^m}$ and polynomials $f_1^{(k)}, f_2^{(k)}, g_1^{(k)}, g_2^{(k)} \in \mathbb{Z}_p^{\leqslant 2^m}[\mathsf{X}]$ for $k \in \{0,1\}$ such that, for each $l \in L$ there exists $\mathsf{ctxt}^{(l)} \in [0, 2^{\ell_{\mathsf{ctxt}}} - 1], \mathsf{cnt}^{(l,0)}, \mathsf{cnt}^{(l,1)} \in [0, N-1]$ where with $x^{(l,b)} = \mathsf{ctxt}^{(b)} 2^{\ell_{\mathsf{cnt}}} + \mathsf{cnt}^{(l,b)}$ for $b \in \{0,1\}$

$$\begin{aligned}
&\mathsf{cnt}^{(l,1)} \cdot \boldsymbol{\gamma}_{0,l} + \boldsymbol{\gamma}_{1,l} - \mathsf{cnt}^{(l,0)} \cdot \boldsymbol{\gamma}_0^{(0)} + \boldsymbol{\gamma}_1^{(0)} \\
&= (g_i^{(0)}(x^{(l,0)}) - g_i^{(1)}(x^{(l,1)}) + \sum_{j\in[2]} T_{i,j}^{(0)}[f_j^{(0)}(x^{(l,0)}) \mod 2^m] - T_{i,j}^{(1)}[f_j^{(1)}(x^{(l,1)}) \mod 2^m])_{i\in[2]} \,.
\end{aligned}$$

Note that $\boldsymbol{\gamma}_0^{(0)}, \boldsymbol{\gamma}_1^{(0)} \in \mathbb{Z}_p^2$ are the same over all runs. For intuition, this corresponds to the fact that the serial numbers $\mathsf{sn}_0 = \mathsf{sn}_1$ in every successful run.

Note that the event Bad is defined over the choice of $\boldsymbol{\gamma}_{0,l}, \boldsymbol{\gamma}_{1,l}$ for $l \in [K_{\max}]$. Here, for a fixed $L \subseteq [K_{\max}]$ of size K, $\boldsymbol{T}_{1,1}^{(k)}, \boldsymbol{T}_{1,2}^{(k)}, \boldsymbol{T}_{2,1}^{(k)}, \boldsymbol{T}_{2,2}^{(k)} \in \mathbb{Z}_p^{2^m}$ and polynomials $f_1^{(k)}, f_2^{(k)}, g_1^{(k)}, g_2^{(k)} \in \mathbb{Z}_p^{\leqslant 2^m}[\mathsf{X}]$ for $k \in \{0,1\}$, and $\mathsf{ctxt}^{(l)} \in [0, 2^{\ell_{\mathsf{ctxt}}} - 1], \mathsf{cnt}^{(l,0)}, \mathsf{cnt}^{(l,1)} \in [0, N-1]$, the probability that the above equation being true is p^{-2K}.

Let Succ be the event that at least K of the $K_{\max}$ runs of $\mathcal{A}''$ does not abort. Note that if Bad does not occur, one of the winning condition of $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}$ or $\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}$ will be satisfied. Hence,

$$\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}, \lambda) \geqslant \Pr[\mathsf{Succ}] - \Pr[\mathsf{Bad}] \,.$$

Adversary $\mathcal{A}'^{\mathsf{H}_{\mathsf{lin}}}(\mathsf{inp})$:

parse $(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*, \mathsf{td}) \leftarrow \mathsf{inp}$

$\mathcal{U} \leftarrow \varnothing$; $\mathsf{idx} \leftarrow 0$; $(i^*, j^*) \leftarrow\!\!\$\ \{(i,j) \in [Q_{\mathrm{Iss}}]^2 : i < j\}$

$(\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k = (\mathsf{dbsp}_k, \mathsf{com}_k, \pi_k)_{k\in\{0,1\}})$

$\quad \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{Iss}, \mathsf{H}_{\mathsf{lin}}, \mathsf{H}_{\mathsf{trunc}}}(\mathsf{crs}, \mathsf{pk}_{\mathsf{I}})$

parse $(\pi_{\mathsf{GS}}^{(k)}, \pi_{\mathsf{lin}}^{(k)}, \pi_{\mathsf{trunc}}^{(k)}) \leftarrow \pi_k$ **for** $k \in \{0,1\}$

for $k \in \{0,1\}$ **do**

$\quad \mathbb{x}_k \leftarrow (\mathsf{crs}, r_k, \mathsf{ctxt}_k, \mathsf{sn}_k, \mathsf{dbsp}_k, \mathsf{com}_k)$

$\quad \widetilde{\mathbb{w}}^{(k)} \leftarrow \mathsf{Ext}_{\mathsf{P},\mathsf{GS}}(\mathsf{td}, \mathbb{x}_k, \pi_{\mathsf{GS},k})$

$\quad$ **parse** $\widetilde{\mathbb{w}}^{(k)}$ as witness in $\widetilde{\mathcal{R}}_{\mathsf{tok}}$

$\quad$ **if** $(\mathbb{x}_k, \widetilde{\mathbb{w}}^{(k)}) \notin \widetilde{\mathcal{R}}_{\mathsf{tok}}$ **then return** $\perp$

if $(\forall k \in \{0,1\} : \mathsf{V}(\mathsf{crs}, \mathsf{ctxt}, r_k, \mathsf{sn}_k, \tau_k) = 1) \ \wedge\ r_0 \neq r_1$

$\quad \wedge\ \mathsf{sn}_0 = \mathsf{sn}_1 \ \wedge\ (\mathsf{dbsp}_0 - \frac{\mathsf{dbsp}_0 - \mathsf{dbsp}_1}{r_0 - r_1} r_0) G_1 \notin \mathcal{U} \ \wedge$

$\quad \{(\mathsf{pk}_{\mathsf{User},i}, C_i, C_{\gamma,i})\}_{i\in\{i^*,j^*\}} = \{(\mathsf{pk}_{\mathsf{User}}^{(k)}, C^{(k)}, C_\gamma^{(k)})\}_{k\in\{0,1\}}$

$\quad \wedge\ \neg\mathsf{BadG} \ \wedge\ \neg\mathsf{Coll}_\gamma$ **then**

$\quad\quad$ **return** $(\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k, \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}})$

return $\perp$

Oracle $\mathrm{Iss}(\mathsf{pk}_{\mathsf{User}}, \mathsf{imsg})$:

$\mathsf{idx} \leftarrow \mathsf{idx} + 1$

if $\mathsf{idx} \neq j^*$ **then** $\boldsymbol{\gamma}_{0,\mathsf{cnt}}, \boldsymbol{\gamma}_{1,\mathsf{cnt}} \leftarrow\!\!\$\ \mathbb{Z}_p^2$

else $(\boldsymbol{\gamma}_{0,\mathsf{cnt}}, \boldsymbol{\gamma}_{1,\mathsf{cnt}}) \leftarrow (\boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*)$

$C_{\gamma,\mathsf{idx}} \leftarrow \sum_{j=1}^{2} \gamma_{0,j,\mathsf{idx}} H_j + \gamma_{1,j,\mathsf{idx}} H_{2+j}$

$\sigma \leftarrow\!\!\$\ \mathsf{SPS.S}(\mathsf{sk}_{\mathsf{I}}, (\mathsf{pk}_{\mathsf{User},\mathsf{idx}}, C_{\mathsf{idx}}, C_{\gamma,\mathsf{idx}}))$

return σ

Oracle $\mathsf{H}_{\mathsf{lin}}(x)$:

return $\mathsf{H}_{\mathsf{lin}}(x)$

Oracle $\mathsf{H}_{\mathsf{trunc}}(x)$:

if $T_{\mathsf{trunc}}[x] = \perp$ **then** $T_{\mathsf{trunc}}[x] \leftarrow\!\!\$\ \mathbb{Z}_p$

return $T_{\mathsf{trunc}}[x]$

Adversary $\mathcal{A}''(\mathsf{inp})$:

parse $(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \boldsymbol{\gamma}_0^*, \boldsymbol{\gamma}_1^*, \mathsf{td}) \leftarrow \mathsf{inp}$

$\rho_{\mathcal{A}'} \leftarrow\!\!\$\ \mathcal{R}_{\mathcal{A}'}$; $\mathsf{out} = (\mathsf{ctxt}, (r_k, \mathsf{sn}_k, \tau_k = (\mathsf{dbsp}_k, \mathsf{com}_k, \pi_k, \widetilde{\mathbb{w}}^{(k)})_{k\in\{0,1\}}) \leftarrow\!\!\$\ \mathcal{A}'^{\mathsf{H}_{\mathsf{lin}}}(\mathsf{inp}; \rho_{\mathcal{A}'})$

// $\mathsf{H}_{\mathsf{lin}}$ is simulated via lazy sampling by $\mathcal{A}''$. Let $\boldsymbol{h}$ be the vector of RO outputs.

$((\mathrm{w}_{\mathsf{lin}}^{(k)})_{k\in\{0,1\}}) \leftarrow\!\!\$\ \mathsf{Ext}_{\mathsf{lin}}^{\mathcal{A}'}(\mathsf{inp}, \mathsf{out}, \boldsymbol{h}; \rho_{\mathcal{A}'})$

// For simplicity, assume $\mathsf{Ext}_{\mathsf{lin}}$'s input out is parsed in the format $((\mathbb{x}_k, \pi_{\mathsf{lin},k})_{k\in\{0,1\}}, \mathsf{aux})$.

if $(\mathrm{w}_{\mathsf{lin}}^{(k)})_{k\in\{0,1\}} = \perp$ **then return** $\perp$

parse $((y_j^{(k)}, \mathsf{rand}_{y_j}^{(k)}, {\beta'_{f,j}}^{(k)}, \mathsf{rand}_{\beta'_{f,j}}^{(k)})_{j\in[2]}, (z_i^{(k)}, \mathsf{rand}_{z_i}^{(k)}, {\beta'_{g,i}}^{(k)}, \mathsf{rand}_{\beta'_{g,i}}^{(k)},$

$\quad (t_{j,i}^{(k)}, \mathsf{rand}_{t_{j,i}}^{(k)})_{j\in[2]})_{i\in[3]}, \boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)}, \mathsf{rand}_{\boldsymbol{\gamma}_0}^{(k)}, \mathsf{rand}_{\boldsymbol{\gamma}_1}^{(k)}) \leftarrow \mathrm{w}_{\mathsf{lin}}^{(k)}$ **for** $k \in \{0,1\}$

parse $\widetilde{\mathbb{w}}^{(k)}$ as witness in $\widetilde{\mathcal{R}}_{\mathsf{tok}}$ **for** $k \in \{0,1\}$

for $k \in \{0,1\}$ **do**

$\quad \mathsf{cnt}^{(k)} \leftarrow \mathrm{dlog}_{G_1} \widehat{\mathsf{cnt}}_1^{(k)}$ // If this takes too long, $\mathcal{A}''$ just aborts.

$\quad$ **for** $j \in [2] : \mathsf{open}_{f,j}^{(k)} \leftarrow ({\beta'_{f,j}}^{(k)}, Q_{f,j}^{(k)})$; $\overline{y}_j^{(k)} \leftarrow \mathrm{dlog}_{G_1} \overline{Y}_{1,j}^{(k)}$

$\quad$ **for** $i \in [2] : \mathsf{open}_{g,i}^{(k)} \leftarrow ({\beta'_{g,i}}^{(k)}, Q_{g,i}^{(k)})$

if $\exists k \in \{0,1\} : \mathsf{cnt}^{(k)} \notin [0, N-1] \ \vee\ (\exists j \in [2] : y_j^{(k)} \neq \overline{y}_j^{(k)} \pmod 2^m \ \vee\ \overline{y}_j^{(k)} \notin [0, 2^m - 1])$

$\quad$ **then return** $\perp$

return $(\mathsf{ctxt}, (C^{(k)}, \mathsf{cnt}^{(k)}, (y_j^{(k)}, \overline{y}_j^{(k)}, \mathsf{open}_{f,j}^{(k)})_{j\in[2]},$

$\quad (t_{1,j}^{(k)}, t_{2,j}^{(k)}, z_j^{(k)}, \mathsf{open}_{T,1,j}^{(k)}, \mathsf{open}_{T,2,j}^{(k)}, \mathsf{open}_{g,j}^{(k)})_{j\in[2]}, \boldsymbol{\gamma}_0^{(k)}, \boldsymbol{\gamma}_1^{(k)})_{k\in[2]})$

Fig. 21. Wrapper adversary $\mathcal{A}'$ and $\mathcal{A}''$ for proof of Lemma 6.8. The highlighted portion denotes the distinction from the proof of Lemma 6.7. Denote $\mathcal{R}_{\mathcal{A}'}$ as the randomness space of $\mathcal{A}'$.

Adversary $\boxed{\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}(\mathsf{crs}_{\mathsf{KZG}})}$, $\boxed{\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}(\mathsf{crs}_{\mathsf{VC}})}$:

parse $\mathsf{par} = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}))$ from $\mathsf{crs}_{\mathsf{KZG}}$

$(\mathsf{crs}_{\mathsf{GS}}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Ext}_{\mathsf{Setup}}(\mathsf{par})$; $\boldsymbol{H} \leftarrow\!\!\$\ \mathbb{G}_1^4$

$\boxed{\mathsf{crs}_{\mathsf{KZG}} \leftarrow\!\!\$\ \mathsf{KZG}_{\mathsf{Ped}}.\mathsf{Setup}(\mathsf{par}, d)}$ $\boxed{\mathsf{crs}_{\mathsf{VC}} \leftarrow\!\!\$\ \mathsf{VC}_{\mathsf{KZG}}.\mathsf{Setup}(\mathsf{par}, 2^m)}$

$\mathsf{crs}_{\mathsf{lin}} \leftarrow\!\!\$\ \Pi_{\mathsf{lin}}.\mathsf{Setup}(\mathsf{crs}_{\mathsf{GS}})$; $\mathsf{crs}_{\mathsf{trunc}} \leftarrow\!\!\$\ \Pi_{\mathsf{trunc}}.\mathsf{Setup}(\mathsf{crs}_{\mathsf{GS}})$

$\mathsf{crs} \leftarrow (\mathsf{par}, \mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{VC}}, \mathsf{crs}_{\mathsf{GS}}, \mathsf{crs}_{\mathsf{lin}}, \mathsf{crs}_{\mathsf{trunc}})$

$(\mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}) \leftarrow\!\!\$\ \mathsf{SPS}.\mathsf{KeyGen}(\mathsf{par})$

$\rho_{\mathcal{A}''} \leftarrow\!\!\$\ \mathcal{R}_{\mathcal{A}''}$ // The random coins contain the index i^* as well.

$\mathsf{Out} \leftarrow \varnothing$

for $l = 1, \ldots, K_{\max}$ **do** // Rewind $\mathcal{A}''$ enough times.

  $\mathsf{out}_l \leftarrow \mathcal{A}''(\mathsf{crs}, \mathsf{sk}_{\mathsf{I}}, \mathsf{pk}_{\mathsf{I}}, \mathsf{td}, \boldsymbol{\gamma}_{0,l}, \boldsymbol{\gamma}_{1,l}; \rho_{\mathcal{A}''})$

  if $\mathsf{out}_l \neq \perp$ **then** $\mathsf{Out} \leftarrow \mathsf{Out} \cup \{\mathsf{out}_l\}$

  if $|\mathsf{Out}| \geqslant K$ **then break**

parse $\mathsf{Out} = \{\mathsf{out}^{(l)}\}_{l\in[K]}$

for $l \in [K]$:

  parse $(\mathsf{ctxt}^{(l)}, (C^{(l,k)}, \mathsf{cnt}^{(l,k)}, (y_j^{(l,k)}, \overline{y}_j^{(l,k)}, \mathsf{open}_{f,j}^{(l,k)})_{j\in[2]},$
  $(t_{1,j}^{(l,k)}, t_{2,j}^{(l,k)}, z_j^{(l,k)}, \mathsf{open}_{T,1,j}^{(l,k)}, \mathsf{open}_{T,2,j}^{(l,k)}, \mathsf{open}_{g,j}^{(l,k)})_{j\in[2]})_{k\in[2]}, \gamma_0^{(l,k)}, \boldsymbol{\gamma}_1^{(l,k)}) \leftarrow \mathsf{out}^{(l)}$

  // $C^{(l,k)}$ are the same for all (l,k) by how $\mathcal{A}''$ is defined.

  for $k \in \{0,1\} : x^{(l,k)} \leftarrow \mathsf{ctxt}^{(l)} 2^{\ell_{\mathsf{cnt}}} + \mathsf{cnt}^{(l,k)}$

$\boxed{\begin{array}{l}\mathcal{E}_{f,j}^{(k)}, \mathcal{E}_{g,i}^{(k)}, \mathcal{E}_{T,j,i}^{(k)} \leftarrow \varnothing \textbf{ for } k \in \{0,1\}, j \in [2], i \in [2] \\ \textbf{for } j \in [2], k \in \{0,1\} : \mathcal{E}_{f,j}^{(k)} \leftarrow \{(x^{(l,k)}, y_j^{(l,k)}, \mathsf{open}_{f,j}^{(l,k)})\}_{l\in[K]} \\ \textbf{for } i \in [2], k \in \{0,1\} : \mathcal{E}_{g,i}^{(k)} \leftarrow \{(x^{(l,k)}, z_i^{(l,k)}, \mathsf{open}_{g,i}^{(l,k)})\}_{l\in[K]} \\ \textbf{for } i,j \in [2], k \in \{0,1\} : \mathcal{E}_{T,j,i}^{(k)} \leftarrow \{(\overline{y}_j^{(l,k)}, t_{j,i}^{(l,k)}, \mathsf{open}_{T,j,i}^{(l,k)})\}_{l\in[K]}\end{array}}$

$\boxed{\begin{array}{l}\textbf{if } \exists i,j \in [2], k \in \{0,1\}, (\overline{y}, t, \mathsf{open}), (\overline{y}', t', \mathsf{open}) \in \mathcal{E}_{T,j,i}^{(k)} : \overline{y} = \overline{y}' \ \wedge \ t \neq t' \\ \quad \textbf{then return } (C^{(1,k)}, \overline{y}, (t, \mathsf{open}), (t', \mathsf{open}))\end{array}}$

$\boxed{\begin{array}{l}\textbf{if } (\exists i \in [2], k \in \{0,1\} : \exists S \subseteq \mathcal{E}_{g,i}^{(k)}, |S| = d+2 : (\forall g \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}], \exists (x, z, \mathsf{open}) \in S : g(x) \neq z)) \\ \quad \textbf{then return } (C_{g,i}^{(1,k)}, S) \\ \textbf{if } (\exists j \in [2], k \in \{0,1\} : \exists S \subseteq \mathcal{E}_{f,j}^{(k)}, |S| = d+2 : (\forall g \in \mathbb{Z}_p^{\leqslant d}[\mathsf{X}], \exists (x, z, \mathsf{open}) \in S : g(x) \neq z)) \\ \quad \textbf{then return } (C_{f,j}^{(1,k)}, S)\end{array}}$

return $\perp$

Fig. 22. Description of adversaries $\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}, \mathcal{B}'_{\mathsf{Coll2},\mathsf{VC}}$. The highlighted pseudocode denotes the distinction from the proof of Lemma 6.8. Denote $\mathcal{R}_{\mathcal{A}''}$ as the randomness space of $\mathcal{A}'$.

We can bound $\mathsf{Pr}[\mathsf{Bad}]$ via a union bound over all subsets $L \subseteq [K_{\max}]$ of size K and all possible vector and polynomials and $\mathsf{ctxt}^{(l)}, \mathsf{cnt}_0^{(l)}, \mathsf{cnt}_1^{(l)}$ for $l \in L$. Note that the probability is over the choices of $\boldsymbol{\gamma}_{0,l}$

$$\begin{aligned}\mathsf{Pr}[\mathsf{Bad}] &\leqslant \binom{K_{\max}}{K} \cdot p^{8\cdot 2^m} \cdot p^{8d} \cdot (2^{\ell_{\mathsf{ctxt}}} N^2)^K \cdot p^{-2K} \\ &\leqslant \left(\frac{eK_{\max}\, pN}{Kp^{2-2^{m+4}/K}}\right)^K \leqslant \left(\frac{eK_{\max}\, N}{Kp^{1-2^{m+4}/K}}\right)^K\end{aligned}$$

The second inequality follows from $d \leqslant 2^m$, $\binom{n}{t} \leqslant (en/t)^t$ and $2^{\ell_{\mathsf{ctxt}}} N \leqslant p$. Setting $K = 2^{m+3}\lambda$ and $K_{\max} = 2K/\varepsilon_{\mathsf{SNColl}_2}$. Then, we have $\mathsf{Pr}[\mathsf{Bad}] \leqslant 2^{-2^{m+3}\lambda}$ with $\varepsilon_{\mathsf{SNColl}_2} \geqslant 8eN/p$.

For $\mathsf{Pr}[\mathsf{Succ}]$, a similar argument to the proof of Lemma 6.7 with the Chernoff bound yields, $\mathsf{Pr}[\mathsf{Succ}] \geqslant (1 - 2^{-2^{m+1}\lambda})\varepsilon_{\mathsf{SNColl}_2}/2 \geqslant \varepsilon_{\mathsf{SNColl}_2}/2 - 2^{-2^{m+1}\lambda}$. Thus concluding the proof that

$$\begin{aligned}\mathsf{Pr}[\mathsf{SNColl}_2] &\leqslant 8Q_{\mathrm{Iss}}^2(\mathsf{Pr}[\mathcal{A}'' \text{ does not abort}] + 2\mathsf{Adv}^{\mathsf{sound}}_{\Pi_{\mathsf{trunc}}}(\mathcal{B}_{\mathsf{sound}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda) \\ &\leqslant 8Q_{\mathrm{Iss}}^2(2\mathsf{Pr}[\mathsf{Succ}] + 2^{-2^{m+1}\lambda+1} + \mathsf{Pr}[\mathsf{BadForm}] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda)) \\ &\leqslant 8Q_{\mathrm{Iss}}^2(\mathsf{Adv}^{\mathsf{dbind}}_{\mathsf{KZG}_{\mathsf{Ped}},d}(\mathcal{B}'_{\mathsf{Coll},\mathsf{KZG}}, \lambda) + \mathsf{Adv}^{\mathsf{pbind}}_{\mathsf{VC}_{\mathsf{KZG}},8k}(\mathcal{B}'_{\mathsf{Coll},\mathsf{VC}}, \lambda) + 2^{-2^{m+1}\lambda+2} + \\ &\quad + \mathsf{Pr}[\mathsf{BadForm}] + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog},3}, \lambda))\ .\square\end{aligned}$$

Table of Contents

1 Introduction . . . 1
1.1 Technical Overview . . . 3
2 Preliminaries . . . 7
3 Everlasting Anonymous Rate-Limited Tokens (EARLT) . . . 12
4 Building Blocks . . . 16
5 EARLT Construction from Bilinear Pairings . . . 19
5.1 Key Generation and Issuance . . . 20
5.2 Showing Protocol . . . 20
5.3 Discussion . . . 30
6 Security of EARLT . . . 31
6.1 Anonymity . . . 32
6.2 Unforgeability . . . 33
6.3 Linkability . . . 37
6.4 Exculpability . . . 39
A Security of Pagh-Pagh Function Family . . . 44
B Security of KZG Commitments . . . 45
B.1 Degree-Binding of $\mathsf{KZG}_{\mathsf{Ped}}$ . . . 46
B.2 Security of $\mathsf{VC}_{\mathsf{KZG}}$ . . . 47
C Deferred Linkability Proofs . . . 47
C.1 Proof of Lemma 6.5 . . . 47
C.2 Proof of Lemma 6.6 . . . 49
C.3 Proof of Lemma 6.7 . . . 50
C.4 Proof of Lemma 6.8 . . . 55