Anonymous Reputation Systems with Revocation, Revisited

Ryuya Hayashi[1,3], Shuichi Katsumata[2,3], and Yusuke Sakai[3]

[1] The University of Tokyo, Tokyo, Japan
rhys@iis.u-tokyo.ac.jp
[2] PQShield Ltd, Oxford, U.K.
shuichi.katsumata@pqshield.com
[3] AIST, Tokyo, Japan
yusuke.sakai@aist.go.jp

Abstract. An *anonymous reputation system* (ARS) was first proposed by Blömer, Juhnke, and Kolb (FC, 2015), a protocol similar to group signatures in concept, and its definition has been refined for example by El Kaafarani, Katsumata, and Solomon (FC, 2018). A representative application of an ARS is e-commerce sites, where users are allowed to *anonymously* write reviews on products they have purchased, while also preventing them from *double* reviewing.
In this work, we revisit ARS. Our contributions are threefolds: First, we show that all previous definitions of ARS allow the users' *purchase history* to leak. While users' privacy is being guaranteed through the notion of anonymity, our findings show that this only achieves a weaker form of privacy, contrary to previously believed. Second, we formally define *purchase privacy*, addressing the above shortcoming, and complement previous security models. Along the way, we notice that one of the main entities, the *system manager*, does not play any cryptographically relevant role in the definition of ARS. Effectively, by excluding the system manager from the definition, we are able to simplify previous definitions. Lastly, we propose a generic construction and provide one concrete efficient instantiation based on pairing-based cryptography, requiring only 16 kilobits for a signature.

1 Introduction

1.1 Background

The use of e-commerce (EC) sites for shopping has become exceedingly commonplace, with reviews by consumers playing a significant role in purchase decisions. However, review credibility is compromised when EC site operators can alter them for incentives. To ensure trustworthy reviews, an *Anonymous Reputation System* (ARS), introduced by Blömer, Juhnke, and Kolb [13], was designed. While initially for EC sites, ARS is applicable to any secure reputation system involving user reviews (e.g., Amazon, TripAdvisor, Airbnb, YouTube).

Requirements for ARS. To ensure the efficacy of an ARS, three essential requirements have been considered. Firstly, only users who have purchased the product should be eligible to submit reviews. This measure not only maintains the reliability of reviews but also acts as a deterrent against potential misuse, such as fake reviews or Sybil attacks. Secondly, user privacy is paramount within the ARS framework. This has been captured by the notion *user anonymity*, informally guaranteeing that users' reviews do not leak the information of the user. Finally, to preserve the credibility of the reputation system, users should be constrained to submit only one review per product. This restriction is crucial because, under the cloak of anonymity, malicious users might submit similar reviews repeatedly, falsely inflating the appearance of widespread consensus among purchasers.

Model and Security Definitions. The formal definition of an ARS, initially proposed by Blömer et al. [13], comprises four entities: the system manager (EC site operator), key issuers[4] (vendors), users, and the tracing manager. A user first registers via the system manager on an EC site; then purchases products from key issuers (vendors); and lastly receives a token from the key issues, granting them the privilege to submit a review for the acquired product. Users with tokens can generate review signatures[5] that are publicly verifiable. Moreover, a tracing manager can detect if some malicious user submitted multiple reviews for the same product, and trace them to the user. A tracing manager can then report this to the authority for removal (i.e., the system manager), preventing malicious users from exploiting anonymity to submit deceptive reviews on different products.

The current five de facto properties of ARS are the following. For the knowledgeable readers, due to the similarity between ARS and group signatures [5, 7, 16], their definitions are close.

- *Unforgeability*: A malicious user cannot sign without receiving a token from the key issuer.
- *Non-frameability*: A malicious user cannot create a signature that is traced to some honest user.
- *Anonymity*: A signature should not reveal the user who signed it, even to the system manager and key issuers.
- *Traceability* and *tracing-soundness*: The tracing manager can trace any valid signature to some user in the system.
- *Public-linkability*: There is a public algorithm that allows checking whether two signatures are created by the same user.

Prior Works. The original definition of Blömer et al. [13] modeled the system and tracing managers to always be honest and lacked the notion of non-frameability and revocation. They constructed an ARS based on the group signature scheme

[4] The name *key issuer* comes from the fact that the vendors are the ones responsible for issuing keys, a.k.a. *tokens*, to the users, allowing them to submit reviews.

[5] Throughout the introduction, we use the phrase "submitting a review" and "generating a signature" interchangeably.

by Boneh, Boyen, and Shacham [14]. Soon after, El Kaafarani, Katsumata, and Solomon [20] showed a shortcoming of their model: it allowed malicious users to falsely link signatures to honest users and sign anonymously. They provided a new definition fixing this issue; however, due to the added complexity of the new definitions, they resorted to treating the system manager and every key issuer to be identical entities. This went against the original motivation of ARS, where the EC site operator and all vendors should be treated differently. They constructed an ARS based on the learning with errors (LWE) assumption. In an independent work, Blömer, Eiden, and Juhnke [12] adopted the UC framework [17] but faced complexity with a five-page ideal functionality and lacked the mechanism to revoke malicious users. Their schemes are only selectively secure, as adversaries must declare the users to be corrupt before the game begins.

1.2 Our Contribution

In this work, we revisit ARS once more and provide a more complete view of the model. In the following, we outline our three main contributions.

Overlooked privacy guarantee: purchase-privacy. While anonymity captures one essential aspect of user privacy, the definition only guarantees that a *signature* does not leak the signer. We observe that this is too weak for ARS as there are other sources of leakage. In an ARS, when a user buys a product and receives a token from the key issuer, the public information maintained by the key issuer is updated.[6] In such a case, the public information may encode information of the users who purchased a specific product sold by the key issuer. Namely, even if a user did not generate a signature (i.e., trivially anonymous), the public information alone may leak whether a user bought the product. Put differently, all prior ARS security models overlook the privacy of users' *purchase history.* It is clearly desirable to consider such privacy as it has been shown that combined with other socially available auxiliary information (e.g., blog posts, social networking sites), purchase history can be further used to trace sensitive information of users, including gender, location, and political preferences [23,31,34,35]. As one example, we can easily attack the purchase history of the ARS construction by El Kaafarani et al. [20] as users' identities are stored in the public information of the key issuer. To address this oversight, we introduce *purchase privacy* to capture this attack formally.

In the realm of group signatures, Backes et al. [3] introduced a similar security notion known as *membership privacy.* It aims to address the scenario where, given two users who have not joined a group, if one of them decides to join, it should be impossible to determine which of the two users joined. However, their definition of membership privacy is very complex, making it difficult to port to an already complex model such as ARS. Instead, our definition of purchase

[6] To be precise, in an incomplete model of ARS where revocation is not considered, we can have a static system parameter. However, if revocation is considered, then the system parameter must encode information of the allowed users.

privacy is straightforward and stated clearly. Informally, we simply modify the syntax so that the membership information related to honest users is not used when updating the public information. To see that this is the correct definition, we show that this simple definition, when combined with anonymity, implies the complex purchase/membership privacy definition by Backes et al. extended to the ARS setting. See Section 4 for a detailed discussion.

New model of ARS. As explained above, El Kaafarani et al. [20] made a conscious choice of handling the system manager and every key issuer as the same entity in their security definition to tame the complexity of the model. In this work, we come back to the original motivation of ARS and treat these entities independently while maintaining the complexity under control. The main insight to keep the complexity minimal is noticing that the system manager in prior ARS models plays a role that is orthogonal to the security offered by ARS. In previous works, the system manager's sole role was to issue certificates to the users upon registering to the reputation system. The only purpose of this certificate was for the users to prove to a key issuer that the they are valid user of the reputation system. Notice this is not an inherent security definition for ARS. Indeed, the same functionality can be easily replicated at the API level. For example, a system manager can maintain a PKI (outside the ARS model), and users can talk to the key issuer via an authenticated channel. By excluding the system manager from the ARS ecosystem, we are able to simplify the security model. It is worth highlighting that since we treat each key issuer independently, unlike in El Kaafarani et al. [20], we can consider a stronger security model where a set of key issuers can act maliciously. We provide the formal model in Section 3.

Generic construction and instantiations. Lastly, we show a generic construction of an ARS that is secure in our new security model. Our approach builds upon the ideas introduced in prior works [5, 20, 27, 28, 37], and we further extend it to satisfy purchase privacy. Our generic construction relies on the integration of various standard cryptographic primitives, including a PKE scheme, a signature scheme, an accumulator, a NIZK proof system, and a linkable indistinguishable tag system [10]. Our definition is general enough to capture accumulators based on the subset difference method [32] using vector commitments [28]. As a concrete example, in this work, we show an efficient instantiation based on pairing with only 16 kilobits for a signature. We note that due to the lack of a practical NIZK proof system for accumulators from lattices, we leave it as an open problem to efficiently instantiate our generic construction via lattices. See Sections 5 and 6 for a detailed discussion.

Independent and concurrent work. Recently, Blömer, Bobolz, and Porzenheim [11] proposed a generic construction of ARS. They provide a generic construction similar to ours and provide a full description of a lattice-based ARS. In contrast, our main contribution lies in the definitional work of ARS and the introduction of purchase-privacy. Moreover, while their ARS model (and hence their construction) lacks the ability to revoke malicious users, ours does not.

2 Preliminaries

Notations are provided in the supplemental material A. This paper uses the following cryptographic ingredients without introducing their definitions. We defer these formal definitions to the supplemental materials A.1, A.2, and A.3.

- an IND-CPA secure public key encryption scheme $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$,
- an EUF-CMA secure signature scheme $\mathsf{SIG} = (\mathsf{SIG.KG}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$,
- a non-interactive zero-knowledge proof $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ that is sound, zero-knowledge, and simulation extractable.

2.1 Linkable Indistinguishable Tag.

We introduce a linkable indistinguishable tag scheme (LIT) that creates a unique tag corresponding to a user, building upon Bernhard et al.'s construction [10]. While the existing LIT definition is for symmetric key settings, we extend it to a public key setting. This extension includes the key checking algorithm, additional security properties (*key-secrecy* and *key-robustness*), as well as tag linking and tag checking algorithms. The following provides a detailed syntax and security properties for LIT.

Definition 2.1. *A linkable and indistinguishable tag system* (LIT) *with an item space* $\mathbb{I}$ *and a tag space* $\mathbb{T}$ *consists of the following PPT algorithms.*

$\mathsf{LIT.KG}(1^n) \to (\mathsf{tagpk}, \mathsf{tagsk})$: *The tag key generation algorithm, given a security parameter* 1^n*, outputs a public tag key* tagpk *and a secret tag key* tagsk.

$\mathsf{LIT.Tag}(\mathsf{tagsk}, I) \to \tau$: *The tag generation algorithm, given a secret tag key* tagsk *and an item* $I \in \mathbb{I}$*, outputs a tag* $\tau \in \mathbb{T}$.

$\mathsf{LIT.Link}(\tau_0, \tau_1) \to 1/0$: *The tag linking algorithm, given two tags* τ_0 *and* τ_1*, outputs* 1 *(linked) or* 0 *(not linked).*

$\mathsf{LIT.ChkKey}(\mathsf{tagpk}, \mathsf{tagsk}) \to 1/0$: *The key checking algorithm, given a public tag key* tagpk *and a secret tag key* tagsk*, outputs* 1 *(accept) or* 0 *(reject).*

$\mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}, I, \tau) \to 1/0$: *The tag checking algorithm, given a secret tag key* tagsk*, an item* $I \in \mathbb{I}$*, and a tag* $\tau \in \mathbb{T}$*, outputs* 1 *(accept) or* 0 *(reject).*

We require a LIT to satisfy the following standard properties. We defer each formal definition of the followings to the supplemental material A.4.

Correctness. It requires the following three properties: (1) any honestly generated key pair passes the key checking algorithm, (2) any honestly generated tag passes the tag checking algorithm, and (3) two tags generated by the same key and the same item are always linked.

Indistinguishability. Any two tags (generated with different keys) for the same item are indistinguishable.

Linkability. Any two tags (generated with the same key) for the same item are linkable.

Key-Secrecy. No can recover the secret key from a public key and tags.

Key-Robustness. If two tags for the same item are linked, then their corresponding secret tag keys are the same.

2.2 Accumulator with Revocation

In this section, we introduce *accumulators with revocation*, which is a generalization of membership revocation proposed by Libert et al. [28]. While prior works [4, 15] proposed definitions for general accumulators, our syntax differs from theirs. We choose to define it in a way that is amenable to our modular construction of ARS. The setup algorithm takes as input the bit length of the accumulated elements and outputs a public parameter and a secret key. The secret key is used to issue a "witness" for an element (we call this an ID) $\mathsf{id} \in \{0,1\}^m$, which is used to produce a proof of inclusion. Namely, only a party in possession of a witness for id can prove the inclusion of id in a given accumulator. The accumulation algorithm is ordinary. It takes as input a set of IDs still not included in an accumulator and outputs an accumulator and auxiliary information, used by the users with a witness to generate a proof. The proof generation algorithm uses an auxiliary information, an ID, and a witness to generate a proof of inclusion. The verification algorithm uses an accumulator, an ID, and a proof to verify if ID is accumulated in the accumulator.

This syntax fits nicely into an ARS. A prover needs to know a witness for the ID to prove inclusion of an accumulator. Looking ahead, this witness is issued by a key issuer to user during the joining protocol.

Definition 2.2. *An accumulator with revocation* ACC *consists of the following PPT algorithms.*

$\mathsf{ACC.Setup}(1^n, m) \to (\mathsf{pp_{acc}}, \mathsf{sk_{acc}})$: *The setup algorithm, given a security parameter* 1^n *and a bit length* m *of IDs, outputs a public parameter* $\mathsf{pp_{acc}}$ *and a secret key* $\mathsf{sk_{acc}}$.

$\mathsf{ACC.Wit}(\mathsf{pp_{acc}}, \mathsf{sk_{acc}}, \mathsf{id}) \to \mathsf{wit_{id}}$: *The witness generation algorithm, given a public parameter* $\mathsf{pp_{acc}}$*, a secret key* $\mathsf{sk_{acc}}$*, and an ID* id*, outputs a witness* $\mathsf{wit_{id}}$ *of the membership of* id.

$\mathsf{ACC.Acc}(\mathsf{pp_{acc}}, R) \to (\mathsf{acc}, \mathsf{aux_{acc}})$: *The accumulation algorithm, given a public parameter* $\mathsf{pp_{acc}}$ *and a set* $R \subseteq \{0,1\}^m$ *of revoked IDs, outputs an accumulator* acc *and auxiliary information* $\mathsf{aux_{acc}}$.

$\mathsf{ACC.Prove}(\mathsf{pp_{acc}}, \mathsf{aux_{acc}}, \mathsf{id}, \mathsf{wit_{id}}) \to \pi$: *The proving algorithm, given a public parameter* $\mathsf{pp_{acc}}$*, auxiliary information* $\mathsf{aux_{acc}}$*, an identity* id*, and a witness* $\mathsf{wit_{id}}$*, outputs a proof* π *of the membership.*

$\mathsf{ACC.Verify}(\mathsf{pp_{acc}}, \mathsf{acc}, \mathsf{id}, \pi) \to 1/0$: *The verification algorithm, given a public parameter* $\mathsf{pp_{acc}}$*, an accumulator* acc*, an identity* id*, and a witness* π*, outputs a bit* 1 *(accept) or* 0 *(reject).*

We require an accumulator to satisfy the following basic properties. We provide each formal definition of the followings in the supplemental materials A.7.

Correctness. For all active users corresponding to the item at the epoch t, the verification algorithm always returns 1 if a proof of membership is honestly generated.

Soundness. No one can forge a proof of membership for any non-active user corresponding to the item at the epoch t.

Succinctness. The size of proof π is $O(\log m)$ where m is the bit length of IDs.

3 Formalizing Anonymous Reputation System

This section proposes our new formalization of anonymous reputation systems (ARS) based on the previous formalizations [13, 20]. In previous works, there existed four entities: a system manager SM who is an EC site operator; key issuers KI who are vendors that sell products on the EC site; users who buy products; and a tracing manager TM who can trace a user from its review. While Blömer et al. [13] proposed the first model for ARS and its construction, there were several caveats. For example, they did not consider the framing scenario where adversaries try to generate a review that links to another review produced by an honest user. Later, El Kaafarani et al. [20] proposed a comprehensive model and a post-quantum construction. However, due to the complexity of handling traceability and revocations in an already complex security model, they made a conscious choice to merge SM and the KIs as the same entity. While capturing a stronger level of security compared to the definition of Blömer et al., they fail to capture the flexibility of a reputation system where the EC site (i.e., SM) and every vendors (i.e., KI) are distinct entities. For example, in practice, a specific vendor may act maliciously, but this is not captured in their model.

We aim to model an ARS that is more in line with a practical reputation system while keeping the complexity of the definition controlled. Our main insight is that SM *plays no cryptographically relevant role in the security model of ARS*. We provide a more detailed discussion on this point in Section 3.4. Below, we formalize an ARS with only three entities (KIs, users, TM), removing SM. The main source of simplification of our new model comes from the fact that we do not need to consider different corruption states between SM and KIs, i.e., it eliminates the necessity to individually consider each security definition based on which parties are corrupted.

3.1 Syntax

In this section, we give the syntax of ARS. An ARS consists of ten stand-alone algorithms ($\mathsf{RepSetup}, \mathsf{KIgen}, \mathsf{TMgen}, \mathsf{UKgen}, \mathsf{RevokeUser}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link}, \mathsf{Trace}$, Judge), and one pair of interactive algorithms $\langle \mathsf{Join}, \mathsf{Issue} \rangle$.

$\mathsf{RepSetup}(1^n) \rightarrow \mathsf{pp}$: On input a security parameter 1^n, it outputs a public parameter pp.

$\mathsf{KIgen}(\mathsf{pp}) \rightarrow (\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathbb{I}^{\mathsf{ipk}})$: On input a public parameter pp, it outputs a public and secret key pair $(\mathsf{ipk}, \mathsf{isk})$, a private empty registration table $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$, and an item list initialized by $\mathbb{I}^{\mathsf{ipk}} \leftarrow \emptyset$.

$\mathsf{TMgen}(\mathsf{pp}) \rightarrow (\mathsf{tpk}, \mathsf{tsk})$: On input a public parameter pp, it outputs a public and secret key pair $(\mathsf{tpk}, \mathsf{tsk})$.

Below, we assume pp is used by all algorithms and omit it from inputs.

$\mathsf{UKgen}(1^n) \rightarrow (\mathsf{upk}, \mathsf{usk})$: This algorithm is run by a user who wants to join the system. On input the security parameter 1^n, it outputs a public and secret key pair $(\mathsf{upk}, \mathsf{usk})$. We assume the (unique) user public key upk is authenticated by a public key infrastructure (PKI) and made public to the entities in the system. Hereafter, we identify a user by its public key upk.

$\langle \mathsf{Join}(\mathsf{upk}, \mathsf{usk}, \mathsf{ipk}, \mathsf{item}), \mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{upk}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}) \rangle :$ This is an interactive protocol between a user upk and a key issuer KI. If this is the first time that users buy an item from the KI, it sets $t \leftarrow 0$ and $\mathsf{info}_0^{\mathsf{ipk}} \leftarrow \emptyset$. Upon successful completion, the key issuer KI issues a token for the user with respect to the purchased item item and updates the registration table $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$, public information $\mathsf{info}_{t+1}^{\mathsf{ipk}}$ for the next epoch, and $\mathbb{I}^{\mathsf{ipk}} \leftarrow \mathbb{I}^{\mathsf{ipk}} \cup \{\mathsf{item}\}$. In addition, the user upk outputs a signing key ssk. We assume that KI outputs $\perp$ if item was included in $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$ before the protocol execution.

$\mathsf{RevokeUser}(\mathsf{isk}, \mathsf{upk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathsf{info}_t^{\mathsf{ipk}}) \rightarrow (\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathsf{info}_{t+1}^{\mathsf{ipk}}) :$ On input KI's secret key isk, an active user upk to be revoked, the registration table $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$, and the current system information $\mathsf{info}_t^{\mathsf{ipk}}$, it outputs an updated registration table $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$ and a new system information $\mathsf{info}_{t+1}^{\mathsf{ipk}}$ for the next epoch $t+1$.

$\mathsf{Sign}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{ssk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}) \rightarrow \Sigma :$ On input a key issuer's public key ipk, the tracing manager's public key tpk, a signing key ssk, system information $\mathsf{info}_t^{\mathsf{ipk}}$ (including epoch t), an item, and a message M, it outputs a signature Σ.

$\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma) \rightarrow 1/0 :$ On input a key issuer's public key ipk, the tracing manager's public key tpk, system information $\mathsf{info}_t^{\mathsf{ipk}}$ (including epoch t), a message M, and a signature Σ, it outputs 1 (accept) or 0 (reject).

$\mathsf{Link}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, (\mathsf{M}_0, \Sigma_0), (\mathsf{M}_1, \Sigma_1)) \rightarrow 1/0 :$ On input a key issuer's public key ipk, the tracing manager's public key tpk, system information $\mathsf{info}_t^{\mathsf{ipk}}$ (including epoch t), an item, and two valid message-signature pairs (M_0, Σ_0) and (M_1, Σ_1), it outputs 1 (accept) or 0 (reject), indicating whether the signatures for item were produced by the same user or not.

$\mathsf{Trace}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{tsk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma) \rightarrow (\mathsf{upk}, \Pi_{\mathsf{Trace}})/\perp :$ On input a key issuer's public key ipk, the tracing manager's key pair $(\mathsf{tpk}, \mathsf{tsk})$, system information $\mathsf{info}_t^{\mathsf{ipk}}$ (including epoch t), an item, a message M, and a signature Σ, it outputs the user upk who produced Σ and a proof Π_{Trace} that attests to this fact. If tracing fails, it outputs $\perp$.

$\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) \rightarrow 1/0 :$ On input a key issuer's public key ipk, a tracing manager's public key tpk, $(\mathsf{tpk}, \mathsf{tsk})$, system information $\mathsf{info}_t^{\mathsf{ipk}}$ (including epoch t), an item, a message M, a signature Σ, a user public key upk, and a tracing proof Π_{Trace}, it outputs 1 (accept) or 0 (reject), indicating whether Π_{Trace} is a valid proof that upk produced Σ or not.

We say that an ARS is correct if the Verify algorithm always accepts reviews produced by honest, non-revoked users and if the honest tracing manager can always identify the signer of such signatures where the Judge algorithm will accept his decision. Additionally, two reviews on the same item produced by the same user should always link.

3.2 New Security Definition: Purchase Privacy

This section defines *purchase privacy*. Intuitively, purchase privacy ensures that the products bought by the user (i.e., purchase history) remains private. To be more precise, the fact that a user bought some product item should only be learnt by the key issuer (i.e., the vendor) selling item. While this seems to be a fundamental security notion for ARS, it turns out that this has never been incorporated in prior works. For example, we found an attack on purchase privacy against El Kaafarani et al. [20]. In the $\langle\mathsf{Join}, \mathsf{Issue}\rangle$ protocol of [20], KI includes the public keys of joined users to the public information, thus trivially leaking user information. In fact, even if the public information was somehow secured, there are other issues. For users to prove that they are not revoked, KI provides a witness for the maintained accumulator. Since [20] uses Merkel trees as the accumulator, each user learns the co-path from its node to the root, effectively including a value of the sibling's public key. Namely, once a user buys some item, it may implicitly learn who else bought it. This illustrates the subtlety of purchase privacy.

To prevent such attacks, we introduce a formal definition of purchase privacy. Our definition is simple: we say an ARS has purchase privacy if the syntax satisfies the following property. We later show why this is the right definition by showing equivalence between membership privacy, a similar (but more complex) security definition proposed by [3] in the context of group signatures.

Definition 3.1 (Purchase Privacy). *An ARS satisfies* purchase-privacy *if it satisfies the following conditions.*

- *The* $\langle\mathsf{Join}, \mathsf{Issue}\rangle$ *protocol can be divided into four parts as follows:*

$$\begin{aligned}
&\mathsf{Join}_1(\mathsf{upk}, \mathsf{usk}, \mathsf{ipk}, \mathsf{item}) \to \rho_1,\\
&\mathsf{Issue}_1(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}) \to \mathsf{info}_{t+1}^{\mathsf{ipk}},\\
&\mathsf{Issue}_2(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}, \rho_1) \to (\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \rho_2),\\
&\mathsf{Join}_2(\mathsf{gpk}, \mathsf{upk}, \mathsf{usk}, \rho_2) \to \mathsf{ssk},
\end{aligned}$$

  where ρ_1 *and* ρ_2 *are messages from the user to the key issuer and from the key issuer to the user, respectively.*
- *The* $\mathsf{RevokeUser}$ *algorithm can be replaced as follows:*

$$\mathsf{RevokeUser}'(\mathsf{isk}, \mathsf{upk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}], \mathsf{info}_t^{\mathsf{ipk}}) \to (\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}], \mathsf{info}_{t+1}^{\mathsf{ipk}}),$$

  where $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$ *is a row* upk *of the table* $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$*, and the other rows of the table are never referred and modified.*

Remark 3.1. In other seemingly related cryptographic protocols, such as those used for anonymous payments [9], there exists a comparable concept known as *unlinkability*. Unlinkability is a security notion designed to ensure privacy for concealing purchase information. However, this concept does not directly apply to ARS as it requires public-linkability to detect double reviewing.

3.3 Basic Security Definitions

In this section, we provide six basic security properties for our ARS. Due to space limitations, we defer each formal definition in the supplemental materials B.1 and B.2.

Anonymity. This ensures that, for any PPT adversary, it is infeasible to distinguish between two reviews produced by two honest reviewers (upk_0, upk_1) of its choice.

Non-Frameability. This ensures that, for any PPT adversary, it is infeasible to forge a valid review that traces or links to an uncorrupted user.

Traceability. This ensures that the (honest) manager of an ARS is always able to trace an active user who produces a valid signature.

Unforgeability. This ensures that an adversary cannot forge a valid review for an item on bealf of an active member managed by an (honest) key issuer.

Tracing Soundness. This ensures that, for any PPT adversary, it is infeasible to output a review that traces back to two different reviewers.

Public-Linkability. This ensures that, for any (possibly inefficient) adversary, it is infeasible to output two reviews for the same item that trace to the same user but do not link.

3.4 Discussion on the Security Model

As mentioned at the beginning of this section, while the previous definition of ARS considered four types of entities (SM, KI, user, TM), we remove SM from the model. The rationale behind this design choice is that the security guarantees provided by SM in the previous definitions can be easily handled at the API level and, in particular, do not hold any cryptographic relevance. Namely, the SM only handles the registration of users to the reputation system. For example, in the original construction of [13], SM issues certificates to users during registration, primarily used to verify their legitimate enrollment with the reputation system. However, this function can be outsourced to a PKI maintained by the system manager (outside of the ARS model), simplifying the process by relying on an authenticated channel between the user and the KI.

More formally, recall the three requirements for an ARS we mentioned in Section 1.1. In the following, we take a closer look at what SM previously did and argue that these three requirements remain intact even if we remove SM.

1. The first requirement was that no user can post reviews for products (i.e., item) they have not purchased. Unforgeability prevents malicious users from creating reviews without a token from an honest KI. While a SM can detect malicious attempts, this can be handled practically at the API level.
2. The second requirement was that purchase information never leaks from reviews and other public system information. Anonymity makes honest user reviews indistinguishable, and purchase privacy prevents KI from disclosing buyers' product details. These security measures remain effective even in the presence of a malicious SM, rendering its role unnecessary.

3. The final requirement was that anyone could link two reviews on the same product by the same user. However, in all previous definitions, this is achieved by the public-linkability property, which in particular is independent of SM.

4 Relation with Membership Privacy

This section compares the purchase privacy defined in Section 3.2 and the membership privacy proposed by [3]. Backes et al. proposed membership privacy, a new security definition for group signature schemes. Roughly, a group signature scheme satisfies membership privacy if membership information does not leak from joined users' signatures and public information.[7] We can easily extend their definition to the ARS setting (and we denote such a definition by BHS-PP). However, BHS-PP is highly complicated, as shown in the supplemental material C.1. To gain confidence in our simple definition of purchase privacy, we show that if there exists an ARS scheme satisfying anonymity and our purchase privacy, then we can construct another ARS scheme that satisfies BHS-PP. This means that there is no need to mind complicated security for purchase privacy as long as it satisfies some syntactical restrcition.[8]

Concretely, we construct an ARS scheme ARS' that satisfies the BHS-PP security assuming an ARS scheme ARS is anonymous and purchase-private. We construct ARS' to have at least one (dummy) member for each item. Adding a dummy member allows us to respond to queries to the challenge oracle in the security game of BHS-PP using the challenge oracle in that of anonymity. The overview of ARS' is as follows: when the key generation algorithm KIgen' is run, it generates a dummy user $(\mathsf{upk}', \mathsf{usk}')$ and appends upk' (resp., usk') to the public (resp., secret) key for the key issuer; when the $\langle \mathsf{Join}', \mathsf{Issue}' \rangle$ protocol is run, if there is no joined user to the group, then it first makes the dummy user upk' join the group and runs the $\langle \mathsf{Join}, \mathsf{Issue} \rangle$ protocol. The remaining algorithms of ARS' are the same as those of ARS. Details are provided in the supplemental material C.2.

Here, we only provide the theorem and defer its proof in the supplemental material C.3.

Theorem 4.1. *If* ARS *is anonymous and purchase private, then there exists an* ARS' *satisfying BHS-PP security.*

5 Our Generic Construction for ARS

Here we provide the intuition of the construction. Our ARS can be regarded as multiple groups of group signatures. There exist multiple key issuers KI, and each KI has a unique key pair $(\mathsf{ipk}, \mathsf{isk})$ and manages a set of group signature

[7] [3] divided the definition into two parts: *join privacy* and *leave privacy*. However, we do not need to consider *leave privacy* since no honest users are revoked in ARS.

[8] As an independent interest, this observation can apply to group signature schemes.

schemes, where each group corresponds to an item sold by the key issuer. There exists only one tracing manager TM corresponding to all key issuers. Every user in the system is registered to the authentication system and has its own key pair $(\mathsf{upk}, \mathsf{usk})$.[9] A user upk who bought an item sold by a KI becomes a member of the group corresponding to the item managed by the KI. When a user joins the group, KI assigns an identity id to the user, accumulates id, and issues $\mathsf{wit}_{\mathsf{id}}$ and θ, indicating that the user is not revoked and id is honestly issued by the KI, respectively. The user receives $\mathsf{wit}_{\mathsf{id}}$ and generates a proof of the membership π_{acc} which indicates that the user bought the item from the KI. After that, the user can write a review for the bought item. Every review has to include the following three parts: the first one is a ciphertext $C = (c_1, c_2)$ of its public key encrypted by the key of TM; the second one is a tag τ with respect to the pair of the KI's identifier and the purchased item, using the user's secret key; and the last one is a proof π indicating that (1) the user has a certificate θ of the membership, (2) the user has a proof π_{acc} to show that it is not revoked at the epoch, and (3) C, τ, θ, and π_{acc} are correctly generated. To anonymously write a review, we use a NIZKPoK to prove a witness about the above three by the relation ρ_1 defined as follows:

$$\begin{aligned}\rho_1 = \{(((c_1,c_2), (\mathsf{vk}, \mathsf{pp}_{\mathsf{acc}}), (\mathsf{ek}_1, \mathsf{ek}_2), \mathsf{acc}, \mathsf{item}, \tau), (\mathsf{upk}, \mathsf{usk}, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2)) \mid \\ c_1 = \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1) \wedge c_2 = \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2) \\ \wedge\ \mathsf{LIT.ChkKey}(\mathsf{upk}, \mathsf{usk}) = 1 \\ \wedge\ \mathsf{LIT.ChkTag}(\mathsf{upk}, \mathsf{usk}, \langle \mathsf{vk}, \mathsf{pp}_{\mathsf{acc}}, \mathsf{item}\rangle, \tau) = 1 \\ \wedge\ \mathsf{ACC.Verify}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{acc}, \mathsf{id}, \pi_{\mathsf{acc}}) = 1 \\ \wedge\ \mathsf{SIG.Ver}(\mathsf{vk}, \langle \mathsf{upk}, \mathsf{id}, \mathsf{item}\rangle, \theta) = 1\}.\end{aligned}$$

A different point from a group signature is that it includes a tag. The tag is used for public linkability, i.e., if the user posts two reviews to the same item sold by the same KI, anyone can detect them by using the $\mathsf{LIT.Link}$ algorithm. Since every signature includes a ciphertext of the signer's public key, the tracing manager TM can decrypt it and learn who is the signer. When TM traces a review, TM outputs not only the signer's public key but also its proof, so it provides a NIZK for the relation ρ_2 defined as follows:

$$\begin{aligned}\rho_2 = \{((\mathsf{ek}_1, c_1, \mathsf{upk}), (\mathsf{dk}_1, r_{\mathsf{PKE}})) \mid \\ \mathsf{upk} = \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1) \wedge (\mathsf{ek}_1, \mathsf{dk}_1) = \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})\}.\end{aligned}$$

5.1 Our Construction

We show our generic construction for ARS from PKE, SIG, NIZK_1 (resp., NIZK_2) for the relation ρ_1 (resp., ρ_2), LIT, and ACC defined in Section 2. Refer to the above for an overview of our construction.

[9] We note again that we assume there exists an authentication scheme behind our ARS and each user has a unique identifier upk.

$\mathsf{RepSetup}(1^n)$: On input a security parameter 1^n, it outputs a public parameter $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$, where $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$.

$\mathsf{KIgen}(\mathsf{pp})$: The key issuer KI runs $(\mathsf{pp}_{\mathsf{acc}}, \mathsf{sk}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Setup}(1^n, n)$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KG}(1^n)$, and sets $\mathsf{ipk} := (\mathsf{vk}, \mathsf{pp}_{\mathsf{acc}})$, $\mathsf{isk} := (\mathsf{sk}, \mathsf{sk}_{\mathsf{acc}})$, $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}} := \emptyset$ and $\mathbb{I}^{\mathsf{ipk}} := \emptyset$. It then outputs $(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathbb{I}^{\mathsf{ipk}})$.

$\mathsf{TMgen}(\mathsf{pp}) \rightarrow (\mathsf{tpk}, \mathsf{tsk})$: The tracing manager TM randomly chooses $r_{\mathsf{PKE}} \leftarrow \{0,1\}^{\mathsf{poly}(n)}$ and computes $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})$ and $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$. It then sets $\mathsf{tpk} := (\mathsf{ek}_1, \mathsf{ek}_2), \mathsf{tsk} := (\mathsf{dk}_1, r_{\mathsf{PKE}})$, and outputs $(\mathsf{tpk}, \mathsf{tsk})$. Note that the second key pair $(\mathsf{ek}_2, \mathsf{dk}_2)$ is not used for the functionality but for the security proof.

$\mathsf{UKgen}(1^n) \rightarrow (\mathsf{upk}, \mathsf{usk})$: The user computes $(\mathsf{tagpk}, \mathsf{tagsk}) \leftarrow \mathsf{LIT.KG}(1^n)$ and sets $\mathsf{upk} := \mathsf{tagpk}$ and $\mathsf{usk} := \mathsf{tagsk}$. Hereafter, the user is identified by his public key upk.

$\langle \mathsf{Join}(\mathsf{upk}, \mathsf{usk}, \mathsf{ipk}, \mathsf{item}), \mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{upk}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}) \rangle$:
A user upk requests to buy a product item at epoch t. The key issuer KI computes in the following five steps:

1. If $t = 0$, then adds $\mathbb{I}^{\mathsf{ipk}} \leftarrow \mathbb{I}^{\mathsf{ipk}} \cup \{\mathsf{item}\}$;
2. If $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$ is empty, then randomly chooses $\mathsf{id}_{\mathsf{upk}} \leftarrow \{\mathsf{id} \in \{0,1\}^n \mid \text{Does not appear in } \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}\}$ and sets $List_{\mathsf{upk}} \leftarrow \epsilon$. Otherwise, let $(\mathsf{id}_{\mathsf{upk}}, \mathtt{status}, List_{\mathsf{upk}}) \leftarrow \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$;
3. If $\mathtt{status} = \mathtt{revoked}$ or $\exists(\mathsf{item}, -) \in List_{\mathsf{upk}}$, then returns $\perp$ and terminates. Otherwise, continues;
4. Sets a set of revoked users' id R as follows:
$R := \{\mathsf{id} \in \{0,1\}^n \mid \exists \mathsf{upk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}] = (\mathsf{id}, \mathtt{revoked}, -)\}$;
5. Computes $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$, $\mathsf{wit}_{\mathsf{id}} \leftarrow \mathsf{ACC.Wit}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{sk}_{\mathsf{acc}}, \mathsf{id})$, $\theta \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \langle \mathsf{upk}, \mathsf{id}, \mathsf{item} \rangle)$, and $\sigma_{\mathsf{acc}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \langle t+1, \mathsf{acc} \rangle)$, where t is included in $\mathsf{info}^{\mathsf{ipk}}_t$.

Finally, it sends $(\mathsf{id}_{\mathsf{upk}}, \mathsf{wit}_{\mathsf{id}}, \theta)$ to the user upk, updates $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}] \leftarrow (\mathsf{id}_{\mathsf{upk}}, \mathtt{active}, List_{\mathsf{upk}} \cup \{(\mathsf{item}, \theta)\})$, and returns $\mathsf{info}^{\mathsf{ipk}}_{t+1} := (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma_{\mathsf{acc}})$. The user upk receives θ from the KI and sets $\mathsf{ssk}[\mathsf{ipk}][\mathsf{item}] := (\mathsf{upk}, \mathsf{usk}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}}, \theta)$.

$\mathsf{RevokeUser}(\mathsf{isk}, \mathsf{upk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}}_t)$: When a key issuer KI revokes a user upk, it revokes upk from all trees it manages in the following five steps:

1. Gets $(\mathsf{id}_{\mathsf{upk}}, -, -) \leftarrow \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$;
2. Sets a set of revoked users' id R as follows:
$R := \{\mathsf{id} \in \{0,1\}^n \mid \exists \mathsf{upk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}] = (\mathsf{id}, \mathtt{revoked}, -)\} \cup \{\mathsf{id}_{\mathsf{upk}}\}$;
3. Computes $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$ and $\sigma_{\mathsf{acc}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \langle t+1, \mathsf{acc} \rangle)$, where t is included in $\mathsf{info}^{\mathsf{ipk}}_t$.

Finally, it updates $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}] \leftarrow (\mathsf{id}_{\mathsf{upk}}, \mathtt{revoked}, -)$ and returns $\mathsf{info}^{\mathsf{ipk}}_{t+1} := (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma_{\mathsf{acc}})$.

$\mathsf{Sign}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{ssk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{item}, \mathsf{M})$: To sign M using $(\mathsf{vk}, \mathsf{pp}_{\mathsf{acc}}) \leftarrow \mathsf{ipk}$, $(\mathsf{ek}_1, \mathsf{ek}_2) \leftarrow \mathsf{tpk}$, $(\mathsf{tagpk}, \mathsf{tagsk}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}}, \theta) \leftarrow \mathsf{ssk}$, and $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma_{\mathsf{acc}}) \leftarrow \mathsf{info}^{\mathsf{ipk}}_t$, it computes in the three steps as follows:

1. Randomly chooses $r_1 \leftarrow \{0,1\}^{\mathsf{poly}(n)}$ and $r_2 \leftarrow \{0,1\}^{\mathsf{poly}(n)}$, and computes $c_1 = \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1)$ and $c_2 = \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2)$. Then, sets $C \leftarrow (c_1, c_2)$;
2. Computes a tag to show that it buys the product item, i.e., $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{tagsk}, \langle \mathsf{vk}, \mathsf{pp}_{\mathsf{acc}}, \mathsf{item} \rangle)$;
3. Computes a proof to show that it is not revoked, i.e., $\pi_{\mathsf{acc}} \leftarrow \mathsf{ACC.Prove}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{aux}_{\mathsf{acc}}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}})$.

Finally, it generates a proof $\pi \leftarrow \mathsf{NIZK.Prove}_1(\mathsf{crs}_1, \mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle, \langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle)$ and outputs $\Sigma := (C, \tau, \pi)$.

$\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma)$: Parse $(C, \tau, \pi) \leftarrow \Sigma$. It returns 1 if $\mathsf{NIZK.Verify}_1(\mathsf{crs}_1, \mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle, \pi) = 1$ and $\mathsf{SIG.Ver}(\mathsf{vk}, \langle t, \mathsf{acc} \rangle, \sigma_{\mathsf{acc}}) = 1$, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma_{\mathsf{acc}}) \leftarrow \mathsf{info}_t^{\mathsf{ipk}}$. Otherwise, it returns 0.

$\mathsf{Link}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, (\mathsf{M}_0, \Sigma_0), (\mathsf{M}_1, \Sigma_1))$: Parse Σ_i as (C_i, τ_i, π_i). If $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}_i, \Sigma_i) = 0$ for $i = 0$ or 1, then it returns 0. Otherwise, it returns the output bit of $\mathsf{LIT.Link}(\tau_0, \tau_1)$.

$\mathsf{Trace}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{tsk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma)$: If $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma) = 0$, then it returns $\perp$. Otherwise, it parses Σ as $((c_1, c_2), \tau, \pi)$ and tsk as $(\mathsf{dk}_1, r_{\mathsf{PKE}})$, and computes $\mathsf{upk} = \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$. If $\mathsf{upk} = \perp$, then it returns $\perp$. Finally, it generates a proof $\Pi_{\mathsf{Trace}} \leftarrow \mathsf{NIZK.Prove}_2(\mathsf{crs}_2, \epsilon, \langle \mathsf{ek}_1, c_1, \mathsf{upk} \rangle, \langle \mathsf{dk}_1, r_{\mathsf{PKE}} \rangle)$, where $(\mathsf{ek}_1, \mathsf{ek}_2) \leftarrow \mathsf{tpk}$, and outputs $(\mathsf{upk}, \Pi_{\mathsf{Trace}})$.

$\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}})$: If $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma) = 0$, then it returns $\perp$. Otherwise, it parses Σ as $((c_1, c_2), \tau, \pi)$ and returns $\mathsf{NIZK.Verify}_2(\mathsf{crs}_2, \epsilon, \langle \mathsf{ek}_1, c_1, \mathsf{upk} \rangle, \Pi_{\mathsf{Trace}})$, where $(\mathsf{ek}_1, \mathsf{ek}_2) \leftarrow \mathsf{tpk}$.

5.2 Security Analysis

We show that our construction is secure. Correctness is straightforward as long as all building blocks are correct. In the following, we prove that our scheme satisfies purchase-privacy defined in Section 3.2. We here provide the theorems with respect to other security properties only, and their proofs are provided in the supplemental materials D.1 to D.6.

Theorem 5.1. *The anonymous reputation system* ARS *is purchase-private.*

This theorem clearly holds with the following reasons:

- In the $\langle \mathsf{Join}, \mathsf{Issue} \rangle$ protocol, to compute $\mathsf{info}_{t+1}^{\mathsf{ipk}} = (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$, where $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$ and $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \langle t+1, \mathsf{acc} \rangle)$, all we need is to pick an unused id, so there is no need to take upk as input.
- In the $\mathsf{RevokeUser}$ algorithm, to compute $\mathsf{info}_{t+1}^{\mathsf{ipk}} = (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$, it needs to take R as input, which is the set of revoked users. Thus, there is no need to refer the entries of $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$ related to active users.

Theorem 5.2. *If the PKE scheme* PKE *is IND-CPA secure, the NIZK proof system Π_1 is zero-knowledge and simulation-extractable, the NIZK proof system Π_2 is zero-knowledge, and the linkable indistinguishable tag system* LIT *is indistinguishable, then the anonymous reputation system* ARS *is anonymous.*

Theorem 5.3. *If the NIZK proof system Π_1 is zero-knowledge and simulation-extractable, the NIZK proof system Π_2 is sound, and the linkable indistinguishable tag system* LIT *is link-sound and key-secret, then the anonymous reputation system* ARS *is non-frameable.*

Theorem 5.4. *If the signature scheme* SIG *is EUF-CMA secure, the NIZK scheme Π_1 is sound and simulation-extractable, and the accumulator* ACC *is sound, then the anonymous reputation system* ARS *is traceable.*

Theorem 5.5. *If the signature scheme* SIG *is EUF-CMA secure, the NIZK scheme Π_1 is simulation-extractable, the NIZK scheme Π_2 is sound, and the accumulator* ACC *is sound, then the anonymous reputation system* ARS *is unforgeable.*

Theorem 5.6. *If the NIZK scheme Π_2 is sound, then the anonymous reputation system* ARS *is tracing sound.*

Theorem 5.7. *If the NIZK scheme Π_1 is knowledge-sound, the NIZK scheme Π_2 is sound, and the subset covering scheme* SC *is linkable, then the anonymous reputation system* ARS *is publicly linkable.*

6 Efficient Instantiation From Pairings

In this section, we provide an overview of a pairing-based efficient instantiation of our generic construction. We defer our concrete instantiation in the supplemental material E.

Public-Key Encryption. We use the ElGamal encryption scheme [21] from the decision Diffie-Hellman assumption.

Signature. We use the Abe-Groth-Haralambiev-Ohkubo structure-preserving signature scheme [1].

Linkable Indistinguishable Tag. We use a simple construction of linkable indistinguishable tags, where $H\colon \{0,1\}^* \to \mathbb{G}_2$ is a cryptographic hash function modeled as a random oracle.

- $\mathsf{LIT.KG}(1^n)$**.** Compute $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^n)$ and choose $x \leftarrow \mathbb{Z}_p$. Set $\mathsf{tagpk} \leftarrow g^x$ and $\mathsf{tagsk} \leftarrow x$ and output $(\mathsf{tagpk}, \mathsf{tagsk})$.
- $\mathsf{LIT.Tag}(\mathsf{tagsk}, I)$**.** Compute $\tau \leftarrow H(I)^{\mathsf{tagsk}}$ and output τ.
- $\mathsf{LIT.Link}(\tau_0, \tau_1)$**.** Check $\tau_0 = \tau_1$. If it holds, output 1. Otherwise, output 0.
- $\mathsf{LIT.ChkKey}(\mathsf{tagpk}, \mathsf{tagsk})$**.** Check $\mathsf{tagpk} = g^{\mathsf{tagsk}}$. If it holds, output 1. Otherwise, output 0.
- $\mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}, I, \tau)$**.** Check $\mathsf{tagpk} = g^{\mathsf{tagsk}}$ and $\tau = H(I)^{\mathsf{tagsk}}$. If they hold, output 1. Otherwise, output 0.

Accumulator. We use the accumulator abstracted from Libert-Peters-Yung group signature scheme [28].

NIZK for ρ_1 and ρ_2. We use Maurer's generic protocol [30] for linear equations and a standard technique for proving quadratic equations using a Σ protocol for a linear equation (Such a technique is, for example, used by Boneh et al. [14]).

Signature Size. A signature of our scheme is 16 kilobits if we instantiate our construction with the BLS12-381 curve. This includes 16 $\mathbb{G}_1$ elements, 6 $\mathbb{G}_2$ elements, and 15 $\mathbb{Z}_p$ elements. See the supplemental material E for details.

Acknowledgement. We thank Keisuke Hara for engaging in helpful discussions in the early phase of the project. We also thank the anonymous referees for their valuable comments and helpful suggestions. This work was partially supported by JSPS KAKENHI Grant Numbers JP18K18055 and JP23KJ0548.

References

1. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg (Aug 2011) 15, 62
2. Abe, M., Jutla, C.S., Ohkubo, M., Pan, J., Roy, A., Wang, Y.: Shorter QA-NIZK and SPS with tighter security. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 669–699. Springer, Heidelberg (Dec 2019) 20
3. Backes, M., Hanzlik, L., Schneider-Bensch, J.: Membership privacy for fully dynamic group signatures. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2181–2198. ACM Press (Nov 2019) 3, 9, 11, 33
4. Baldimtsi, F., Camenisch, J., Dubovitskaya, M., Lysyanskaya, A., Reyzin, L., Samelin, K., Yakoubov, S.: Accumulators with applications to anonymity-preserving revocation. In: 2017 IEEE European Symposium on Security and Privacy. pp. 301–315. IEEE Computer Society Press (2017) 6
5. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (May 2003) 2, 4
6. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 390–399. ACM Press (Oct / Nov 2006) 25
7. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (Feb 2005) 2
8. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (Apr 2007) 73
9. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press (May 2014) 9
10. Bernhard, D., Fuchsbauer, G., Ghadafi, E., Smart, N., Warinschi, B.: Anonymous attestation with user-controlled linkability. Int. J. Inf. Secur. **12**, 219–249 (2013) 4, 5
11. Blömer, J., Bobolz, J., Porzenheim, L.: A generic construction of an anonymous reputation system and instantiations from lattices. Cryptology ePrint Archive, Paper 2023/464 (2023), https://eprint.iacr.org/2023/464 4

12. Blömer, J., Eidens, F., Juhnke, J.: Practical, anonymous, and publicly linkable universally-composable reputation systems. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 470–490. Springer, Heidelberg (Apr 2018) 3
13. Blömer, J., Juhnke, J., Kolb, C.: Anonymous and publicly linkable reputation systems. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 478–488. Springer, Heidelberg (Jan 2015) 1, 2, 7, 10
14. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (Aug 2004) 3, 15
15. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 561–586. Springer, Heidelberg (Aug 2019) 6
16. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of fully dynamic group signatures. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 16. LNCS, vol. 9696, pp. 117–136. Springer, Heidelberg (Jun 2016) 2
17. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001) 3
18. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (Aug 2006) 20
19. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003), `https://doi.org/10.1137/S0097539702403773` 37
20. El Kaafarani, A., Katsumata, S., Solomon, R.: Anonymous reputation systems achieving full dynamicity from lattices. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 388–406. Springer, Heidelberg (Feb / Mar 2018) 3, 4, 7, 9
21. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (Aug 1984) 15, 61
22. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (Dec 2012) 22
23. Gervais, A., Ritzdorf, H., Lucic, M., Lenders, V., Capkun, S.: Quantifying location privacy leakage from transaction prices. In: Askoxylakis, I.G., Ioannidis, S., Katsikas, S.K., Meadows, C.A. (eds.) ESORICS 2016, Part II. LNCS, vol. 9879, pp. 382–405. Springer, Heidelberg (Sep 2016) 3
24. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (Dec 2006) 22
25. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 581–612. Springer, Heidelberg (Aug 2017) 20
26. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (Dec 2013) 20
27. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 373–403. Springer, Heidelberg (Dec 2016) 4

28. Libert, B., Peters, T., Yung, M.: Group signatures with almost-for-free revocation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 571–589. Springer, Heidelberg (Aug 2012) 4, 6, 15, 65
29. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 499–517. Springer, Heidelberg (Feb 2010) 68
30. Maurer, U.: Zero-knowledge proofs of knowledge for group homomorphisms. Designs, Codes and Cryptography **77**(2-3), 663–676 (Jun 2015) 15, 70, 73
31. Mayer, J.R., Mitchell, J.C.: Third-party web tracking: Policy and technology. In: 2012 IEEE Symposium on Security and Privacy. pp. 413–427. IEEE (2012) 3
32. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (Aug 2001) 4, 24, 65
33. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. pp. 427–437. ACM Press (May 1990) 41
34. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: 2008 IEEE Symposium on Security and Privacy. pp. 111–125. IEEE Computer Society Press (May 2008) 3
35. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: 2009 IEEE Symposium on Security and Privacy. pp. 173–187. IEEE Computer Society Press (May 2009) 3
36. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (Aug 1993) 73
37. Tsudik, G., Xu, S.: Accumulating composites and improved group signing. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (Nov / Dec 2003) 4

Supplemental Materials

A Formal Definitions of Basics

Notation. In this paper, we use the following notations. $x \leftarrow X$ denotes sampling an element x from a finite set X uniformly at random. $y \leftarrow \mathcal{A}(x; r)$ denotes that a probabilistic algorithm $\mathcal{A}$ outputs y for an input x using a randomness r, and we simply denote $y \leftarrow \mathcal{A}(x)$ when we need not write an internal randomness explicitly. For interactive Turing machines $\mathcal{A}$ and $\mathcal{B}$, $(v_a, v_b) \leftarrow \langle \mathcal{A}(x_a), \mathcal{B}(x_b) \rangle$ denotes that $\mathcal{A}$ (resp., $\mathcal{B}$) outputs v_a (resp., v_b) at the end of an execution of an interactive protocol between $\mathcal{A}$ and $\mathcal{B}$, where $\mathcal{A}$ and $\mathcal{B}$ take x_a and x_b as input, respectively. $x \coloneqq y$ denotes that x is defined by y. n denotes a security parameter. A function $f(\lambda)$ is a negligible function in λ if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. $\mathsf{negl}(\lambda)$ denotes an unspecified negligible function. PPT stands for probabilistic polynomial time. $\emptyset$ denotes the empty set. If n is a natural number, $[n]$ denotes the set of integers $\{1, \cdots, n\}$. Also, if a and b are integers such that $a \leq b$, $[a, b]$ denotes the set of integers $\{a, \cdots, b\}$. If $\mathcal{O}$ is a function or an algorithm and $\mathcal{A}$ is an algorithm, $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{A}\{\mathcal{O}\}$ denote that $\mathcal{A}$ has oracle access to $\mathcal{O}$.

A.1 Public-Key Encryption

Definition A.1 (Public-Key Encryption). *A PKE scheme* PKE *with a plaintext space* $\mathbb{M}$ *consists of the following three PPT algorithms.*

$\mathsf{PKE.KG}(1^n) \to (\mathsf{ek}, \mathsf{dk})$ *: The key generation algorithm, given a security parameter* 1^n*, outputs an encryption key* ek *and a decryption key* dk*.*

$\mathsf{PKE.Enc}(\mathsf{ek}, m) \to \psi$ *: The encryption algorithm, given an encryption key* ek *and a plaintext* m*, outputs a ciphertext* ψ*.*

$\mathsf{PKE.Dec}(\mathsf{dk}, \psi) \to m$ *: The (deterministic) decryption algorithm, given a decryption key* dk*, and a ciphertext* ψ*, outputs a plaintext* $m \in \{\bot\} \cup \mathbb{M}$*.*

We require a PKE scheme to satisfy the following standard properties: correctness and IND-CPA security.

Definition A.2 (Correctness). *A PKE scheme* PKE *satisfies correctness if for all* $n \in \mathbb{N}$ *and* $m \in \mathbb{M}$*, we have*

$$\Pr[(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.KG}(1^n) : \mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{PKE.Enc}(\mathsf{ek}, m)) = m] = 1.$$

Definition A.3 (IND-CPA Security). *We say that a PKE scheme* PKE *is IND-CPA secure if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(n) := \left| \Pr\left[\begin{array}{c} b \leftarrow \{0,1\}, \\ (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.KG}(1^n), \\ (m_0^*, m_1^*, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{ek}), \\ \psi^* \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, m_b^*), \\ b' \leftarrow \mathcal{A}(\psi^*, \mathsf{st}) \end{array} : b = b' \right] - \frac{1}{2} \right|,$$

where it is required that $|m_0^*| = |m_1^*|$ *holds.*

A.2 Signature

Definition A.4 (Signature). *A signature scheme* SIG *with a message space* $\mathbb{M}$ *consists of the following three PPT algorithms.*

$\mathsf{SIG.KG}(1^n) \to (\mathsf{vk}, \mathsf{sk})$ *: The key generation algorithm, given a security parameter* 1^n*, outputs a verification key* vk *and a signing key* sk*.*

$\mathsf{SIG.Sign}(\mathsf{sk}, m) \to \sigma$ *: The signing algorithm, given a signing key* sk *and a message* m*, outputs a signature* σ*.*

$\mathsf{SIG.Ver}(\mathsf{vk}, m, \sigma) \to 1/0$ *: The (deterministic) verification algorithm, given a verification key* vk*, a message* m*, and a signature* σ*, outputs either* 1 *(accept) or* 0 *(reject).*

We require a signature scheme to satisfy the following standard properties: correctness and EUF-CMA security.

Definition A.5 (Correctness). *A signature scheme* SIG *satisfies correctness if for all* $n \in \mathbb{N}$ *and* $m \in \mathbb{M}$*, we have*

$$\Pr[(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KG}(1^n) : \mathsf{SIG.Ver}(\mathsf{vk}, m, \mathsf{SIG.Sign}(\mathsf{sk}, m)) = 1] = 1.$$

Definition A.6 (EUF-CMA Security). *We say that a signature scheme* SIG *is EUF-CMA secure if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG},\mathcal{A}}(n) := \Pr\left[\begin{array}{c} \mathsf{SL} := \emptyset, \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KG}(1^n), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{sign}}(\mathsf{vk}) \end{array} : \begin{array}{c} \mathsf{SIG.Ver}(\mathsf{vk}, m^*, \sigma^*) = 1 \\ \wedge\ m^* \notin \mathsf{SL} \end{array}\right],$$

where the signing oracle $\mathcal{O}_{sign}$ *is defined as follows:*

Signing Oracle. *When* $\mathcal{A}$ *accesses the signing oracle* $\mathcal{O}_{sign}$ *by making a query* m*, it computes* $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m)$*, returns* σ *to* $\mathcal{A}$*, and appends* m *to* SL*.*

Definition A.7. *A signature scheme* SIG *is a strongly unforgeable one-time signature scheme if for all PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KG}(1^n), \\ (m, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{vk}), \\ \sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}(\sigma, \mathsf{state}), \\ : (m^*, \sigma^*) \neq (m, \sigma) \wedge \mathsf{SIG.Ver}(\mathsf{vk}, m^*, \sigma^*) = 1 \end{array}\right] = \mathsf{negl}(n).$$

A.3 Non-Interactive Zero-Knowledge Proof

We define a non-interactive zero-knowledge proof of knowledge protocol (or simply NIZK). Below, we define a variant where the proof is generated with respect to a label [2, 26]. Although syntactically different, such NIZK is similar to the notion of signature of knowledge [18, 25].

Definition A.8 (NIZK proof system). *Let* $\mathbb{L}$ *denote a label space where checking membership can be done efficiently. A non-interactive zero-knowledge (*NIZK*) proof system* Π *for a NP relation* $\mathcal{R}$ *consists of oracle-calling PPT algorithms* $(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ *defined as follows:*

$\mathsf{NIZK.Setup}(1^n) \rightarrow \mathsf{crs}$: *The setup algorithm, given a security parameter* 1^n*, outputs a common reference string* crs*.*

$\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \mathsf{W}) \rightarrow \pi$: *The prove algorithm, given a common reference string* crs*, outputs a proof* π*.*

$\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \pi) \to 1/0$: *The verification algorithm, given a common reference string* crs*, a label* $\mathsf{lbl} \in \mathbb{L}$*, a statement* X*, and a proof* π*, outputs either* 1 *(accept) or* 0 *(reject).*

We require a NIZK to satisfy the following standard properties: correctness, soundness, and zero-knowledge defined as follows.

Definition A.9 (Correctness). *A* NIZK Π *satisfies correctness if for all* $n \in \mathbb{N}$*,* $\mathsf{lbl} \in \mathbb{L}$*,* $(\mathsf{X}, \mathsf{W}) \in \mathcal{R}$*, we have*

$$\Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^n), \\ \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \mathsf{W}) \end{array} : \mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \pi) = 1\right] = 1.$$

Definition A.10 (Soundness). *We say that a* NIZK Π *is* sound *if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{sound}}_{\Pi,\mathcal{A}} := \Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^n), \\ (\mathsf{lbl}, \mathsf{X}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array} : \begin{array}{c} \mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \pi) = 1 \\ \wedge\ \mathsf{X} \notin L_{\mathcal{R}} \end{array}\right],$$

where $L_{\mathcal{R}}$ *is the NP language such that* $L_{\mathcal{R}} := \{x \mid \exists w, (x, w) \in \mathcal{R}\}$*.*

Definition A.11 (Zero-knowledge). *Let* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *be a zero-knowledge simulator for* Π*. We say that a* NIZK Π *is* zero-knowledge *if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{A}} := \left|\begin{array}{c} \Pr[\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^n) : \ \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1] \\ -\Pr[(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_0(1^n) : \mathcal{A}^{\mathcal{S}(\mathsf{crs},\tau,\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1] \end{array}\right|,$$

where $\mathcal{P}$ *and* $\mathcal{S}$ *are oracles that on input* $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *return* $\perp$ *if* $\mathsf{lbl} \notin \mathbb{L} \vee (\mathsf{X}, \mathsf{W}) \notin \mathcal{R}$ *and otherwise return* $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *or* $\mathsf{Sim}_1(\mathsf{crs}, \tau, \mathsf{lbl}, \mathsf{X})$*, respectively.*

In addition, we require a NIZK to satisfy simulation extractability, informally meaning that there exists an extractor SimExt with respect to a zero-knowledge simulator such that it can extract a witness from any valid proof.

Definition A.12 (Simulation extractability). *We say that a* NIZK Π *is* simulation extractable *(with respect to zero-knowledge simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$*), if there exists a PPT extractor* $\mathsf{SimExt} = (\mathsf{SimExt}_0, \mathsf{SimExt}_1)$ *such that no PPT adversary* $\mathcal{A}$ *has non-negligible advantage, where* $\mathcal{A}$*'s advantage is defined as follows:*

$$\mathsf{Adv}^{\mathsf{se}}_{\Pi,\mathcal{A}} := \Pr\left[\begin{array}{c} (\mathsf{crs}, \tau, \xi) \leftarrow \mathsf{SimExt}_0(1^n), \\ (\mathsf{lbl}, \mathsf{X}, \pi) \leftarrow \mathcal{A}^{\mathsf{Sim}_1(\mathsf{crs},\tau,\cdot,\cdot)}(\mathsf{crs}), \\ \mathsf{W} \leftarrow \mathsf{SimExt}_1(\mathsf{crs}, \xi, \mathsf{lbl}, \mathsf{X}, \pi) \end{array} : \begin{array}{c} \mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \pi) = 1 \\ \wedge\ (\mathsf{X}, \mathsf{W}) \notin \mathcal{R} \\ \wedge\ (\mathsf{lbl}, \mathsf{X}, \pi) \notin L_{\mathcal{S}} \end{array}\right],$$

where $L_{\mathcal{S}}$ *is the list of simulation queries and responses* $(\mathsf{lbl}_i, \mathsf{X}_i, \pi_i)_i$*, and* SimExt_0 *outputs are identical to* Sim_0 *when restricted to the first two outputs.*

Remark A.1 (Other formalization). Although Groth [24] defines a stronger notion of simulation extractability where ξ can be provided to $\mathcal{A}$, we only require the weaker variant where ξ is kept hidden from $\mathcal{A}$. Our generic construction naturally works if we relied on NIZKs in the random oracle model (ROM) [22]. We chose the definition in the standard model since the formal definition of simulation extractability in the ROM is notoriously contrived, and we believe this helps our generic construction to be more readable. For the interested readers, we emphasize that the usual definition of simulation extractability where the extractor can program the random oracle and rewind the adversary (as in Fiat-Shamir NIZKs) suffices for our generic construction.

A.4 Linkable Indistinguishable Tag

We require a LIT to satisfy the following standard properties: correctness, meaning that (1) any honestly generated key pair passes the key checking algorithm, (2) any honestly generated tag passes the tag checking algorithm, and (3) two tags generated by the same key and the same item are always linked; indistinguishability, any two tags (generated with different keys) for the same item are indistinguishable; linkability, meaning that any two tags generated with the same public tag key for the same item are always linked; key-secrecy, meaning that no one can forge a secret key corresponding to the given public key; and key-robustness, meaning that, if two tags for the same item are linked, then their corresponding secret tag keys are the same.

Definition A.13 (Correctness). *A linkable indistinguishable tag scheme* LIT *satisfies correctness if for all $n \in \mathbb{N}$ and $I \in \mathbb{I}$, we have*

$$\Pr\left[\begin{array}{rl} (\mathsf{tagpk}, \mathsf{tagsk}) \leftarrow \mathsf{LIT.KG}(1^n), & \mathsf{LIT.Link}(\tau_0, \tau_1) = 1 \\ \tau_0 \leftarrow \mathsf{LIT.Tag}(\mathsf{tagsk}, I), & : \wedge\ \mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}, I, \tau_0) = 1 \\ \tau_1 \leftarrow \mathsf{LIT.Tag}(\mathsf{tagsk}, I) & \wedge\ \mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}, I, \tau_1) = 1 \end{array}\right] = 1.$$

Definition A.14 (Indistinguishability). *A linkable indistinguishable tag scheme* LIT *is* indistinguishable *if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{tag\text{-}ind}}_{\mathsf{LIT},\mathcal{A}}(n) := \left| \Pr\left[\begin{array}{c} \mathsf{IL} := \emptyset,\ b \leftarrow \{0,1\}, \\ (\mathsf{tagpk}_0, \mathsf{tagsk}_0) \leftarrow \mathsf{LIT.KG}(1^n), \\ (\mathsf{tagpk}_1, \mathsf{tagsk}_1) \leftarrow \mathsf{LIT.KG}(1^n), \\ (I^*, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ind}}_{tag}}(\mathsf{tagpk}_0, \mathsf{tagpk}_1), \\ \tau^* \leftarrow \mathsf{LIT.Tag}(\mathsf{tagsk}_b, I^*), \\ b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ind}}_{tag}}(\tau^*, \mathsf{st}) \end{array} : b = b' \right] - \frac{1}{2} \right|,$$

where $\mathcal{A}$ cannot query I^* to the tag generation oracle $\mathcal{O}^{\mathsf{ind}}_{tag}$, which is defined as follows:

Tag Generation Oracle. *When $\mathcal{A}$ accesses the tag generation oracle $\mathcal{O}_{tag}^{\mathsf{ind}}$ by making a query (d, I), it computes $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{tagsk}_d, I)$, returns τ to $\mathcal{A}$.*

Definition A.15 (Linkability). *A linkable indistinguishable tag scheme* LIT *satisfies* linkability *if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}_{\mathsf{LIT},\mathcal{A}}^{\mathsf{link}}(n) := \Pr\left[\begin{array}{c}(\mathsf{tagpk}, \mathsf{tagsk}_0, \mathsf{tagsk}_1, I, \tau_0, \tau_1) \leftarrow \mathcal{A}(1^n) : \\ \forall i \in \{0,1\}, \mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}_i, I, \tau_i) = 1 \\ \wedge\ \mathsf{LIT.Link}(\tau_0, \tau_1) = 0\end{array}\right].$$

Definition A.16 (Key-Secrecy). *A linkable indistinguishable tag scheme* LIT *satisfies* key-secrecy *if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}_{\mathsf{LIT},\mathcal{A}}^{\mathsf{key\text{-}sec}}(n) := \Pr\left[\begin{array}{c}(\mathsf{tagpk}, \mathsf{tagsk}) \leftarrow \mathsf{LIT.KG}(1^n), \mathsf{tagsk}^* \leftarrow \mathcal{A}^{\mathcal{O}_{tag}^{\mathsf{sec}}}(\mathsf{tagpk}) : \\ \mathsf{LIT.ChkKey}(\mathsf{tagpk}, \mathsf{tagsk}^*) = 1\end{array}\right],$$

where the tag generation oracle $\mathcal{O}_{tag}^{\mathsf{sec}}$ is defined as follows:

Tag Generation Oracle. *When $\mathcal{A}$ accesses the tag generation oracle $\mathcal{O}_{tag}^{\mathsf{sec}}$ by making a query I, it computes $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{tagsk}, I)$, returns τ to $\mathcal{A}$.*

Definition A.17 (Key-Robustness). *A linkable indistinguishable tag scheme* LIT *satisfies* key-robustness *if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}_{\mathsf{ATS},\mathcal{A}}^{\mathsf{key\text{-}robust}}(n) := \Pr\left[\begin{array}{c}(I, \mathsf{tagpk}_0, \mathsf{tagsk}_0, \tau_0, \mathsf{tagpk}_1, \mathsf{tagsk}_1, \tau_1) \leftarrow \mathcal{A}(1^n) : \\ \forall i \in \{0,1\}, \mathsf{LIT.ChkTag}(\mathsf{tagpk}_i, \mathsf{tagsk}_i, I, \tau_i) = 1 \\ \wedge\ \mathsf{LIT.Link}(\tau_0, \tau_1) = 1 \\ \wedge\ \mathsf{tagsk}_0 \neq \mathsf{tagsk}_1\end{array}\right].$$

A.5 Vector Commitment

We define vector commitments. A vector commitment scheme allows us to commit to a vector of messages. The scheme then allows to open a coordinate of the vector *compactly*, that is, the size of an opening is sublinear in the dimension of the vector. The scheme needs to satisfy *position binding*, which ensures that we cannot open the same position to two different ways.

The formal definition is as follows.

Definition A.18. *A vector commitment scheme consists of the following PPT algorithms.*

$\mathsf{VC.Setup}(1^n, q) \rightarrow \mathsf{ck}$: *The setup algorithm, given a security parameter 1^n and a dimension $q = \mathsf{poly}(n)$, outputs a commitment key* ck.

$\mathsf{VC.Com}(\mathsf{ck}, (m_1, \ldots, m_q)) \to (\mathsf{com}, \mathsf{aux})$: *The committing algorithm, given a commitment key* ck *and a message vector* $(m_1, \ldots, m_q)$*, outputs a commitment* com *and auxiliary information* aux*.*

$\mathsf{VC.Open}(\mathsf{ck}, \mathsf{aux}, i) \to \mathsf{open}$: *The opening algorithm, given a commitment key* ck*, auxiliary information* aux*, and an index* $i \in [q]$*, outputs an opening* open*.*

$\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, i, m, \mathsf{open}) \to 1/0$: *The verification algorithm, given a commitment key* ck*, a commitment* com*, an index* i*, a message* m*, and an opening* open*, outputs* 1 *(accept) or* 0 *(reject).*

We require VC to satisfy the following two properties: correctness and position-binding.

Definition A.19 (Correctness). *A vector commitment scheme* VC *is correct if for any* n*,* $q \in \mathbb{N}$*,* $(m_1, \ldots, m_q) \in (\{0,1\}^*)^q$*, and* $i \in [q]$ *it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{ck} \leftarrow \mathsf{VC.Setup}(1^n, q), \\ (\mathsf{com}, \mathsf{aux}) \leftarrow \mathsf{VC.Com}(\mathsf{ck}, (m_1, \ldots, m_q)), \\ \mathsf{open} \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \mathsf{aux}, i) : \\ \mathsf{VC.Ver}\left(\mathsf{ck}, \mathsf{com}, i, m_i, \mathsf{open}\right) = 1 \end{array}\right] = 1.$$

Definition A.20 (Position Binding). *A vector commitment scheme* VC *is position binding if for any* $q \in \mathbb{N}$ *and any PPT* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{pos\text{-}bind}}_{\mathsf{VC},\mathcal{A}}(n) := \Pr\left[\begin{array}{c} \mathsf{ck} \leftarrow \mathsf{VC.Setup}(1^n, q), \\ (\mathsf{com}, i, m, \mathsf{open}, m', \mathsf{open}') \leftarrow \mathcal{A}(\mathsf{ck}) : \\ \mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, i, m, \mathsf{open}) = 1 \\ \wedge\, \mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, i, m', \mathsf{open}') = 1 \\ \wedge\, m \neq m' \end{array}\right].$$

A.6 Subset Difference Method

In this paper, we use the subset difference method [32]. While the subset difference method is proposed as a combinatorical tool for broadcast encryption, we formalize this technique as one for expressing a subset of $\{0,1\}^\ell$ compactly. More concretely, the subset difference method is given a subset $R \subseteq \{0,1\}^\ell$ which is *not* included in the subset to be expressed and outputs a set of subsets S_{p_1,s_1}, $\ldots$, S_{p_r,s_r} whose union $S_{p_1,s_1} \cup \cdots \cup S_{p_r,s_r}$ is equal to $\{0,1\}^\ell \setminus R$. Here, each subset $S_{p,s}$ is described by p, $s \in \{0,1\}^{\leq \ell}$ where p is a prefix of s and has a form of

$$S_{p,s} = \{x \in \{0,1\}^\ell \mid p \text{ is a prefix of } x \wedge s \text{ is not a prefix of } x\}.$$

Naor et al. [32] showed that there is an algorithm $\mathsf{SD}(1^\ell, R)$ that outputs a set of subsets $((p_1, s_1), \ldots, (p_r, s_r))$ which satisfy that $S_{p_1,s_1} \cup \cdots \cup S_{p_r,s_r} = \{0,1\}^\ell \setminus R$ and $r \leq 2r - 1$.

A.7 Accumulator

We provide the formal definitions of ACC for correctness, soundness, and succinctness.

Definition A.21. *An accumulator with revocation scheme* ($\mathsf{ACC.Setup}, \mathsf{ACC.Wit}, \mathsf{ACC.Acc}, \mathsf{ACC.Prove}, \mathsf{ACC.Verify}$) *satisfies correctness if for all* $n \in \mathbb{N}$, $m \in \mathbb{N}$, *all* $\mathsf{id} \in \{0,1\}^m$, $R \subseteq \{0,1\}^m$ *such that* $\mathsf{id} \notin R$, *it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{pp_{acc}} \leftarrow \mathsf{ACC.Setup}(1^n, m), \\ \mathsf{wit_{id}} \leftarrow \mathsf{ACC.Wit}(\mathsf{pp_{acc}}, \mathsf{sk_{acc}}, \mathsf{id}), \\ (\mathsf{acc}, \mathsf{aux_{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp_{acc}}, R), \\ \pi \leftarrow \mathsf{ACC.Prove}(\mathsf{pp_{acc}}, \mathsf{aux_{acc}}, \mathsf{id}, \mathsf{wit_{id}}), \\ : \mathsf{ACC.Verify}(\mathsf{pp_{acc}}, \mathsf{acc}, \mathsf{id}, \pi) = 1 \end{array}\right] = 1.$$

Definition A.22. *An accumulator with revocation scheme* ($\mathsf{ACC.Setup}, \mathsf{ACC.Wit}, \mathsf{ACC.Acc}, \mathsf{ACC.Prove}, \mathsf{ACC.Verify}$) *satisfies soundness if for all PPT adversary* $\mathcal{A}$ *and all* $m \in \mathbb{N}$, *it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{pp_{acc}} \leftarrow \mathsf{ACC.Setup}(1^n, m), \\ (R, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{wit}}}(\mathsf{pp_{acc}}), \\ (\mathsf{acc}, \mathsf{aux_{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp_{acc}}, R), \\ (\mathsf{id}^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{wit}}}(\mathsf{state}, \mathsf{acc}, \mathsf{aux_{acc}}), \\ : \mathsf{id}^* \in R \wedge \mathsf{ACC.Verify}(\mathsf{pp_{acc}}, \mathsf{acc}, \mathsf{id}^*, \pi^*) = 1 \end{array}\right] = \mathsf{negl}(n).$$

where the oracle $\mathcal{O}_{\mathrm{wit}}$, *given an identity* $\mathsf{id} \in \{0,1\}^m$, *returns* $\mathsf{wit_{id}} \leftarrow \mathsf{ACC.Wit}(\mathsf{pp_{acc}}, \mathsf{sk_{acc}}, \mathsf{id})$.

Definition A.23. *An accumulator with revocation scheme* ($\mathsf{ACC.Setup}, \mathsf{ACC.Wit}, \mathsf{ACC.Acc}, \mathsf{ACC.Prove}, \mathsf{ACC.Verify}$) *is succinct if a proof* π *output by* $\mathsf{ACC.Prove}$ *has size* $O(\log m)$.

A.8 General Forking Lemma

To prove our instantiation secure, we use the forking lemma by Bellare and Neven [6].

Lemma A.1 (General Forking Lemma). *Fix an integer $Q \geq 1$ and a set H of size $h \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that on input x, h_1, …, h_Q returns a pair, the first element of which is a integer in the range 0, …, Q and the second element of which we refer to as a side output. Let IG be a randomized algorithm that we call the input generator. The accepting probability of $\mathcal{A}$, denoted acc, is defined as the probability that $J \geq 1$ in the experiment*

$$x \leftarrow \mathsf{IG}; h_1, \ldots, h_Q \leftarrow H; (J, \sigma) \leftarrow A(x, h_1, \ldots, h_Q).$$

The forking algorithm $\mathcal{F}_{\mathcal{A}}$ associated to $\mathcal{A}$ is the randomized algorithm that takes input x proceeds as follows:

```
Algorithm $\mathcal{F}_{\mathcal{A}}(x)$:
    Pick coins $\mathsf{coin}$ for $\mathcal{A}$ at random
    $h_1, \ldots, h_Q \leftarrow H$
    $(I, \sigma) \leftarrow \mathcal{A}(x, h_1, \ldots, h_Q; \mathsf{coin})$
    if $I = 0$ then return $(0, \varepsilon, \varepsilon)$
    $h'_I, \ldots, h'_Q \leftarrow H$
    $(I'\sigma') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_Q; \mathsf{coin})$
    if $I = I'$ and $h_I \neq h'_I$ then return $(1, \sigma, \sigma')$
    else return $(0, \varepsilon, \varepsilon)$.
```

Let

$$\mathsf{frk} = \Pr[x \leftarrow \mathsf{IG}; (b, \sigma, \sigma') \leftarrow \mathcal{F}_{\mathcal{A}}(x) : b = 1].$$

Then

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}}{Q} - \frac{1}{h}\right).$$

Alternatively,

$$\mathsf{acc} \leq \frac{Q}{h} + \sqrt{Q \cdot \mathsf{frk}}.$$

A.9 Pairing Groups

We call $\mathcal{G}$ a pairing group generator if $\mathcal{G}$, given a security parameter 1^n, outputs $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ where p is a prime, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are multiplicative gruops of order p, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \leftarrow \mathbb{G}_T$ is a non-degenerated bilinear map, and g and h are respectively generators of $\mathbb{G}$ and $\mathbb{H}$.

We use the following hardness assumption.

Definition A.24. *The symmetric q-Diffie-Hellman exponent assumption for $\mathcal{G}$ holds if for all PPTs $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathbb{G}(1^n), \\ \\ \alpha \leftarrow \mathbb{Z}_p, \\ \\ : \mathcal{A}(\mathsf{gk}, (g^{\alpha^i})_{i \in [2q] \setminus \{q+1\}}, (h^{\alpha^i})_{i \in [2q] \setminus \{q+1\}}) = h^{\alpha^{q+1}} \end{array}\right] = \mathsf{negl}(n).$$

Definition A.25. *The decisional Diffie-Hellman (DDH) assumption in $\mathbb{G}_1$ for $\mathcal{G}$ holds if for all PPTs $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{G},\mathcal{A}} := \left| \Pr\left[\begin{array}{c} \mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathbb{G}(1^n), \\ \alpha,\ \beta,\ \gamma \leftarrow \mathbb{Z}_p, \\ : \mathcal{A}(\mathsf{gk}, g^\alpha, g^\beta, g^\gamma) = 1 \end{array}\right] - \Pr\left[\begin{array}{c} \mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathbb{G}(1^n), \\ \alpha,\ \beta \leftarrow \mathbb{Z}_p,\ \gamma \leftarrow \alpha\beta \\ : \mathcal{A}(\mathsf{gk}, g^\alpha, g^\beta, g^\gamma) = 1 \end{array}\right] \right| = \mathsf{negl}(n).$$

Definition A.26. *The discrete logarithm (DL) assumption in $\mathbb{G}_1$ for $\mathcal{G}$ holds if for all PPTs $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathbb{G}(1^n), \\ \alpha \leftarrow \mathbb{Z}_p, \\ : \mathcal{A}(\mathsf{gk}, g^\alpha) = \alpha \end{array}\right] = \mathsf{negl}(n).$$

B Basic security properties of ARS

B.1 Helpful Oracles for Defining Security

Here, we define several oracles to aid the description of the security experiments. Each oracle may internally store some lists and we provide an overview of these lists in Table 1. It is helpful to keep in mind that the oracles are named in such a way that roughly $\mathsf{SndTo}\underline{\mathsf{X}}$ means the adversary $\mathcal{A}$ engages in a protocol execution with an honest X, and $\mathsf{SndTo}\underline{\mathsf{X}}\text{-}\underline{\mathsf{Y}}$ means $\mathcal{A}$ invokes a protocol execution between an honest X and Y.

$\mathsf{AddU}()$: It generates $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathsf{UKgen}(1^n)$. If it outputs $\mathsf{upk} \in \mathsf{HUL}$, then it outputs $\perp$. [10] Otherwise, it outputs the user public key upk and updates $\mathsf{HUL} \leftarrow \mathsf{HUL} \cup \{\mathsf{upk}\}$.

$\mathsf{AddKI}()$: It generates $(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathbb{I}^{\mathsf{ipk}}) \leftarrow \mathsf{KIgen}(\mathsf{pp})$. It outputs $\perp$ if $\mathsf{ipk} \in \mathsf{HKIL}$. Otherwise, it outputs the key issuer's public key ipk and updates $\mathsf{HKIL} \leftarrow \mathsf{HKIL} \cup \{\mathsf{ipk}\}$.

[10] This restriction is required for the assumption that each user has a unique public key upk.

Table 1: Lists used by oracles and security experiments

Lists	Elements
HUL	Honest users
HKIL	Honest key issuers
HSSKL	Honest system secret keys for users
SL	All signatures that are generated by the oracle OSign
CL	All signatures that are generated by the oracle $\mathsf{AnonChal}_b$ and $\mathsf{PurChalSign}_b$
$\mathsf{IsActive}$	Activation state of a user in the system

$\mathsf{SndToU}(\mathsf{ipk}, \mathsf{upk}, \mathsf{item})$: It outputs $\perp$ if $\mathsf{ipk} \notin \mathsf{HUL}$ or $\mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}] \neq \perp$. Otherwise, it executes $\langle \mathsf{Join}(\mathsf{upk}, \mathsf{usk}, \mathsf{ipk}, \mathsf{item}), \mathcal{A}\rangle$, where it plays the role of the honest user. If Join outputs a signing key ssk, then it updates $\mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}] := \mathsf{ssk}$.

$\mathsf{SndToKI}(\mathsf{ipk}, \mathsf{upk}, \mathsf{item})$: It outputs $\perp$ if $\mathsf{ipk} \notin \mathsf{HKIL}$. Otherwise, it executes $\langle \mathcal{A}, \mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}}_{t^{\mathsf{ipk}}_{\mathsf{cur}}}, \mathsf{upk}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}})\rangle$, where it plays the role of the honest key issuer. If Issue outputs an updated $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}$ and public information $\mathsf{info}^{\mathsf{ipk}}_{t^{\mathsf{ipk}}_{\mathsf{cur}}+1}$ for the next epoch $t^{\mathsf{ipk}}_{\mathsf{cur}} + 1$, then it outputs $\mathsf{info}^{\mathsf{ipk}}_{t^{\mathsf{ipk}}_{\mathsf{cur}}+1}$ and updates $\mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}] := [0, \infty]$.

$\mathsf{SndToKIU}(\mathsf{ipk}, \mathsf{upk}, \mathsf{item})$: It outputs $\perp$ if $\mathsf{ipk} \notin \mathsf{HKIL}$ or $\mathsf{upk} \notin \mathsf{HUL}$ or $\mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}] \neq \perp$. Otherwise, it executes $\langle \mathsf{Join}(\mathsf{upk}, \mathsf{usk}, \mathsf{ipk}, \mathsf{item}), \mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}}_{t^{\mathsf{ipk}}_{\mathsf{cur}}}, \mathsf{upk}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}})\rangle$, where it plays the role of the honest user and the honest key issuer. If Join outputs a signing key ssk, then it updates $\mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}] := \mathsf{ssk}$. If Issue outputs an updated $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}$ and public information $\mathsf{info}^{\mathsf{ipk}}_{t^{\mathsf{ipk}}_{\mathsf{cur}}+1}$ for the next epoch $t^{\mathsf{ipk}}_{\mathsf{cur}} + 1$, then it outputs $\mathsf{info}^{\mathsf{ipk}}_{t^{\mathsf{ipk}}_{\mathsf{cur}}+1}$ and updates $\mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}] := [0, \infty]$.

$\mathsf{AnonChal}_b(\mathsf{ipk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M})$: [11] It checks if the following conditions hold:

- $\mathsf{ssk}_i \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}_i][\mathsf{item}]$ satisfies $\mathsf{ssk}_i \neq \perp$ for $i = 0$ and 1;
- $\exists(\mathsf{ipk}, \mathsf{upk}_i, \mathsf{item}, -, -) \in \mathsf{SL}$ for $i = 0$ or 1.

If not, it outputs $\perp$. Otherwise, it generates a signature $\Sigma_i \leftarrow \mathsf{Sign}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{ssk}_i, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{item}, \mathsf{M})$ and checks if $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{item}, \mathsf{M}, \Sigma_i) = 1$ for $i = 0$ and 1, where $\mathsf{info}^{\mathsf{ipk}}_t$ includes an epoch t. Finally, if all checks pass, it outputs Σ_b and updates $\mathsf{CL} \leftarrow \mathsf{CL} \cup \{(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma_b)\}$.

$\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{upk}, \mathsf{item}, \mathsf{M})$: It checks if the following conditions hold:

- $\mathsf{ssk} = \perp$, where $\mathsf{ssk} \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}]$;
- $\exists(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{CL}$ s.t. $\mathsf{upk} \in \{\mathsf{upk}_0, \mathsf{upk}_1\}$.

[11] This oracle is used in the experiment for anonymity. We assume that in the experiment, an adversary can access this oracle only once.

If not, it outputs $\perp$. Otherwise, it outputs a signature $\Sigma \leftarrow \mathsf{Sign}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{ssk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M})$ and updates $\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(t, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma)\}$, where $\mathsf{info}_t^{\mathsf{ipk}}$ includes an epoch t.

$\mathsf{OTrace}(\mathsf{ipk}, \mathsf{item}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{M}, \Sigma)$: It outputs $\perp$ if there exists $(\mathsf{ipk}, -, -, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{CL}$. Otherwise, it outputs $(\mathsf{upk}, \Pi_{\mathsf{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{tsk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma)$.

$\mathsf{RevUser}(\mathsf{upk})$: It executes $(\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}+1}^{\mathsf{ipk}}) \leftarrow \mathsf{RevokeUser}(\mathsf{isk}, \mathsf{upk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}})$ for all $\mathsf{ipk} \in \mathsf{HKIL}$, using the current registration table $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$ and public information $\mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}}$, and outputs updated $\mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}+1}^{\mathsf{ipk}}$ for the next epoch. It then updates $\mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}] := [0, t_{\mathsf{cur}}^{\mathsf{ipk}}]$.

$\mathsf{RtrKIReg}(\mathsf{ipk})$: It outputs $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$.

B.2 Formal Security Definitions of Basic Properties

Correctness. We say that an ARS is correct if reviews produced by honest, non-revoked users are always accepted by the Verify algorithm and if the honest tracing manager can always identify the signer of such signatures where his decision will be accepted by the Judge algorithm. Additionally, two reviews produced by the same user on the same item should always link. The more precise definition is as follows:

Definition B.1 (Correctness). *[t] An ARS is correct if for any PPT adversary $\mathcal{A}$ involved in the following experiment, we have $\mathsf{Adv}_{\mathsf{ARS},\mathcal{A}}^{\mathsf{correct}}(n) := \Pr[\mathsf{Exp}_{\mathsf{ARS},\mathcal{A}}^{\mathsf{correct}}(n) = 1] = \mathsf{negl}(n)$.*

```
Experiment $\mathsf{Exp}_{\mathsf{ARS},\mathcal{A}}^{\mathsf{correct}}(n)$
pp $\leftarrow$ RepSetup$(1^n)$; HUL, HKIL, HSSKL, IsActive $:= \emptyset$;
(tpk, tsk) $\leftarrow$ TMgen(pp);
(ipk, upk, item, $\mathsf{M}_0, \mathsf{M}_1$) $\leftarrow \mathcal{A}$ {AddU, AddKI, SndToKIU, RevUser, RtrKIReg} (tpk);
if upk $\notin$ HUL or ipk $\notin$ HKIL or HSSKL[ipk][upk][item] $= \perp$
    or $t_{\mathsf{cur}}^{\mathsf{ipk}} \notin$ IsActive[ipk][item][upk] then return 0;
ssk $\leftarrow$ HSSKL[ipk][upk][item];
$\Sigma_i \leftarrow \mathsf{Sign}\left(\mathsf{ipk}, \mathsf{tpk}, \mathsf{ssk}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}_i\right)$ for $i = 0$ and 1;
$(\mathsf{ipk}_i, \Pi_{\mathsf{Trace},i}) \leftarrow \mathsf{Trace}\left(\mathsf{ipk}, \mathsf{tpk}, \mathsf{tsk}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}_i, \Sigma_i\right)$ for $i = 0$ and 1;
if $\mathsf{Verify}\left(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}_i, \Sigma_i\right) = 0$ for $i = 0$ or 1 then return 1;
if $\mathsf{Judge}\left(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}_i, \Sigma_i, \mathsf{upk}_i, \Pi_{\mathsf{Trace},i}\right) = 0$ or upk $\neq \mathsf{upk}_i$
    for $i = 0$ or 1 then return 1;
if $\mathsf{Link}\left(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}}, \mathsf{item}, (\mathsf{M}_0, \Sigma_0), (\mathsf{M}_1, \Sigma_1)\right)$ then return 1;
return 0;
```

Fig. 1: The experiment for defining correctness.

Anonymity. An ARS is anonymous if for any PPT adversary, it is infeasible to distinguish between two reviews produced by two honest reviewers (upk_0, upk_1) of its choice. Below, we only consider the case where the adversary can query the oracle $\mathsf{AnonChal}_b$ once for simplicity.

Definition B.2 (Anonymity). *An ARS is anonymous if we have* $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{ARS},\mathcal{A}}(n) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{anon}\text{-}0}_{\mathsf{ARS},\mathcal{A}}(n) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{anon}\text{-}1}_{\mathsf{ARS},\mathcal{A}}(n) = 1\right]\right| = \mathsf{negl}(n)$ *for any PPT adversary $\mathcal{A}$ involved in the experiment defined in Figure 2.*

```
Experiment $\mathsf{Exp}^{\mathsf{anon}\text{-}b}_{\mathsf{ARS},\mathcal{A}}(n)$
  $\mathsf{pp} \leftarrow \mathsf{RepSetup}(1^n)$; $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} := \emptyset$;
  $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TMgen}(\mathsf{pp})$;
  $d \leftarrow \mathcal{A}\{\mathsf{AddU}, \mathsf{SndToU}, \mathsf{AnonChal}_b, \mathsf{OSign}, \mathsf{OTrace}\}(\mathsf{pp}, \mathsf{tpk})$;
  if $|\mathsf{CL}| \neq 1$ then return $0$;
  return $d$;
```

Fig. 2: The experiment for defining anonymity.

Remark B.1 (Multi-challenge security). We can consider the definition that allows an adversary to query the challenge oracle multiple times. In such a case, the following trivial attacks need to be avoided: Let $(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{CL}$. If $\exists(\mathsf{ipk}, \mathsf{upk}'_0, \mathsf{upk}'_1, \mathsf{item}, \mathsf{M}', \Sigma') \in \mathsf{CL}$ s.t. $\{\mathsf{upk}_0, \mathsf{upk}_1\} \cap \{\mathsf{upk}'_0, \mathsf{upk}'_1\} \neq \emptyset$, then we can identify the challenge bit b by the Link algorithm. In the security definition where we prevent the above attacks from occurring, an adversary must choose a new pair of two honest users every time querying the challenge oracle, so it is easy to reduce the multi-challenge security to the single challenge security via a simple hybrid argument.

Non-Frameability. An ARS is non-frameable if for any PPT adversary, it is infeasible to forge a valid review that traces or links to an uncorrupted user.

Definition B.3 (Non-Frameability). *An ARS is non-frameable if we have* $\mathsf{Adv}^{\mathsf{non}\text{-}\mathsf{frame}}_{\mathsf{ARS},\mathcal{A}}(n) := \Pr\left[\mathsf{Exp}^{\mathsf{non}\text{-}\mathsf{frame}}_{\mathsf{ARS},\mathcal{A}}(n) = 1\right] = \mathsf{negl}(n)$ *for any PPT adversary $\mathcal{A}$ involved in the following experiment.*

```
Experiment $\mathsf{Exp}^{\mathsf{non\text{-}frame}}_{\mathsf{ARS},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(n)$
  $\mathsf{pp} \leftarrow \mathsf{RepSetup}(1^n); \mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} := \emptyset;$
  $(\mathsf{st}, \mathsf{tpk}) \leftarrow \mathcal{A}_1(\mathsf{pp})$
  $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}}) \leftarrow \mathcal{A}_2 \{\mathsf{AddU}, \mathsf{SndToU}, \mathsf{OSign}\} (\mathsf{st});$
  if $\mathsf{upk} \notin \mathsf{HUL}$ then return $0$;
  if $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 0$ or $(-, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{SL}$ then
      return $0$;
  if $\exists(-, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}', \Sigma') \in \mathsf{SL}$
      s.t. $\mathsf{Link}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, (\mathsf{M}, \Sigma), (\mathsf{M}', \Sigma')) = 1$ then return $1$;
  if $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 1$ then return $1$;
  return $0$;
```

Fig. 3: The experiment for defining non-frameability.

Traceability. Traceability ensures that the (honest) manager of an ARS is always able to trace an active user who produces a valid signature. In the following, we give the definition of traceability for a single honest key issuer. The case where the adversary can add multiple key issuers is implied by the single honest key issuer case by a union bound.

Definition B.4 (Traceability). *An ARS is traceable if we have* $\mathsf{Adv}^{\mathsf{trace}}_{\mathsf{ARS},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{ARS},\mathcal{A}}(n) = 1] = \mathsf{negl}(n)$ *for any PPT adversary* $\mathcal{A}$ *involved in the following experiment.* [12]

```
Experiment $\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{ARS},\mathcal{A}}(n)$
  $\mathsf{pp} \leftarrow \mathsf{RepSetup}(1^n);\ \mathsf{HKIL}, \mathsf{IsActive} := \emptyset;$
  $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TMgen}(\mathsf{pp});$
  $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) \leftarrow \mathcal{A} \{\mathsf{AddKI}, \mathsf{SndToKI}, \mathsf{RevUser}, \mathsf{RtrKIReg}\} (\mathsf{pp}, \mathsf{tpk}, \mathsf{tsk});$
  if $|\mathsf{HKIL}| \neq 1$ then return $0$;
  $\{\mathsf{ipk}\} := \mathsf{HKIL};$
  if $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 0$ then return $0$;
  $(\mathsf{upk}, \Pi_{\mathsf{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{tsk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma);$
  Let $t$ be the epoch included in $\mathsf{info}$;
  if $t \notin \mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}]$ then return $1$;
  if $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 0$ then return $1$;
  return $0$;
```

Fig. 4: The experiment for defining traceability.

Unforgeability. Unforgeability ensures that an adversary cannot forge a valid review for an item as an active member managed by an (honest) key issuer.

[12] Note that in the traceability game, we allow $\mathcal{A}$ to get the (honestly) generated tracing secret key tsk, but not to choose tsk maliciously. If we allow an adversary to choose the key of TM maliciously, it can win the game trivially by choosing just $(\mathsf{tpk}, \mathsf{tsk}) = (\perp, \perp)$ since the algorithm Judge always outputs 0 in this case.

Similar to the definition of traceability, we give the definition of unforgeability for a (single) honest key issuer.

Definition B.5 (Unforgeability). *An ARS is unforgeable if we have* $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{ARS},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\mathsf{unf}}_{\mathsf{ARS},\mathcal{A}}(n) = 1] = \mathsf{negl}(n)$ *for any PPT adversary* $\mathcal{A}$ *involved in the following experiment.*

```
Experiment $\mathsf{Exp}^{\mathsf{unf}}_{\mathsf{ARS},\mathcal{A}}(n)$
  $\mathsf{pp} \leftarrow \mathsf{RepSetup}(1^n)$; $\mathsf{HKIL}, \mathsf{IsActive} := \emptyset$;
  $(\mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) \leftarrow \mathcal{A}\{\mathsf{AddKI}, \mathsf{SndToKI}, \mathsf{RevUser}, \mathsf{RtrKIReg}\}(\mathsf{pp})$;
  if $|\mathsf{HKIL}| \neq 1$ then return $0$;
  $\{\mathsf{ipk}\} := \mathsf{HKIL}$;
  if $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 0$
      or $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 0$ then return $0$;
  Let $t$ be the epoch included in $\mathsf{info}$;
  if $t \notin \mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}]$ then return $1$;
  return $0$;
```

Fig. 5: The experiment for defining unforgeability.

Tracing Soundness. An ARS has tracing soundness if for any PPT adversary, it is infeasible to output a review that traces back to two different reviewers.

Definition B.6 (Tracing Soundness). *An ARS is tracing-sound if we have* $\mathsf{Adv}^{\mathsf{trace\text{-}sound}}_{\mathsf{ARS},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\mathsf{trace\text{-}sound}}_{\mathsf{ARS},\mathcal{A}}(n) = 1] = \mathsf{negl}(n)$ *for any PPT adversary* $\mathcal{A}$ *involved in the following experiment.*

```
Experiment $\mathsf{Exp}^{\mathsf{trace\text{-}sound}}_{\mathsf{ARS},\mathcal{A}}(n)$
  $\mathsf{pp} \leftarrow \mathsf{RepSetup}(1^n)$;
  $(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \{\mathsf{upk}_i, \Pi_{\mathsf{Trace},i}\}_{i=0,1}) \leftarrow \mathcal{A}(\mathsf{pp})$;
  if $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 0$ then return $0$;
  if $\mathsf{upk}_0 = \mathsf{upk}_1$ then return $0$;
  if $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}_i, \Pi_{\mathsf{Trace}_i}) = 0$ for $i = 0$ or $1$ then return $0$;
  return $1$;
```

Fig. 6: The experiment for defining tracing-soundness.

Public-Linkability. An ARS is linkable if for any (possibly inefficient) adversary, it is infeasible to output two reviews for the same item that trace to the same user but does not link. This should hold even if the adversary can choose the keys of SM and TM and the secret keys of key issuers.

Definition B.7 (Public-Linkability). *An ARS is public-linkable if we have* $\mathsf{Adv}^{\mathsf{public\text{-}link}}_{\mathsf{ARS},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\mathsf{public\text{-}link}}_{\mathsf{ARS},\mathcal{A}}(n) = 1] = \mathsf{negl}(n)$ *for any PPT adversary* $\mathcal{A}$ *involved in the following experiment.*

```
Experiment $\mathsf{Exp}^{\mathsf{public\text{-}link}}_{\mathsf{ARS},\mathcal{A}}(n)$
  $\mathsf{pp} \leftarrow \mathsf{RepSetup}(1^n)$;
  $(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{upk}, \{\mathsf{M}_i, \Sigma_i, \Pi_{\mathsf{Trace},i}\}_{i=0,1}) \leftarrow \mathcal{A}(\mathsf{pp})$;
  if $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}_i, \Sigma_i) = 0$ for $i = 0$ or $1$ then return $0$;
  if $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}_i, \Sigma_i, \mathsf{upk}, \Pi_{\mathsf{Trace}_i}) = 0$ for $i = 0$ or $1$ then return $0$;
  if $\mathsf{Link}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, (\mathsf{M}_0, \Sigma_0), (\mathsf{M}_1, \Sigma_1)) = 1$ then return $0$;
  return $1$;
```

Fig. 7: The experiment for defining public-linkability.

C Relation between Membership Privacy and Purchase Privacy

C.1 Definition of Membership Privacy [3]

Firstly, we introduce the helpful oracles for defining BHS-PP in the same way as Section B.1.

$\mathsf{OSign}^*_{\mathsf{upk}_0,\mathsf{upk}_1}(\mathsf{ipk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{upk}, \mathsf{item}, \mathsf{M})$: It outputs $\perp$ if $(-, \mathsf{upk}, -, \mathsf{item}, -, -) \in \mathsf{CL}$ or $(-, -, \mathsf{upk}, \mathsf{item}, -, -) \in \mathsf{CL}$. Otherwise, it works in the same way as $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{upk}, \mathsf{item}, \mathsf{M})$.

$\mathsf{RevUser}^*(R)$: It outputs $\perp$ if $\mathsf{upk}_0 \in R$ or $\mathsf{upk}_1 \in R$. Otherwise, it works in the same way as $\mathsf{RevUser}(R)$.

$\mathsf{PrivChal}_{b,\mathsf{upk}_0,\mathsf{upk}_1}(\mathsf{ipk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{item}, \mathsf{M})$: It works as the same way as $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{upk}_b, \mathsf{item}, \mathsf{M})$ except that it updates $\mathsf{CL} \leftarrow \mathsf{CL} \cup \{(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma)\}$ instead of updating SL.

Next, we introduce the definition of BHS-PP. Note that we consider the definition that allows an adversary to query the challenge oracle only one time. However, the advantage of an adversary in breaking the scheme with multiple challenge queries can be upper bounded by a function of the advantage of an adversary of comparable resources in breaking the scheme with single challenge query via a simple hybrid argument.

Definition C.1 (BHS-PP). *An ARS satisfies BHS-PP security if we have* $\left|\Pr[\mathsf{Exp}^{\mathsf{bhs\text{-}pp\text{-}0}}_{\mathsf{ARS},\mathcal{A}}(n) = 1] - \Pr[\mathsf{Exp}^{\mathsf{bhs\text{-}pp\text{-}1}}_{\mathsf{ARS},\mathcal{A}}(n) = 1]\right| = \mathsf{negl}(n)$ *for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *involved in the following experiment.*

C.2 Construction

We modify an ARS scheme to have at least one (dummy) member for each item. Let $\mathsf{ARS} = (\mathsf{RepSetup}, \mathsf{KIgen}, \mathsf{TMgen}, \mathsf{UKgen}, \langle \mathsf{Join}, \mathsf{Issue} \rangle, \mathsf{RevokeUser}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link}, \mathsf{Trace}, \mathsf{Judge})$. ARS' is constructed as follows:

```
Experiment $\mathsf{Exp}^{\mathsf{bhs\text{-}pp\text{-}}b}_{\mathsf{ARS},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(n)$
    $\mathsf{pp} \leftarrow \mathsf{RepSetup}(1^n); \mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL}, \mathsf{IsActive} := \emptyset;$
    $(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathbb{I}^{\mathsf{ipk}}) \leftarrow \mathsf{KIgen}(\mathsf{pp}); \mathsf{HKIL} := \{\mathsf{ipk}\};$
    $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TMgen}(\mathsf{pp});$
    $(\mathsf{st}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}) \leftarrow \mathcal{A}_1 \left\{ \begin{matrix} \mathsf{AddU}, \mathsf{SndToKI}, \mathsf{SndToKIU}, \\ \mathsf{OSign}, \mathsf{OTrace}, \mathsf{RevUser}^* \end{matrix} \right\} (\mathsf{pp}, \mathsf{ipk}, \mathsf{tpk}, \mathbb{I}^{\mathsf{ipk}});$
    if $\{\mathsf{upk}_0, \mathsf{upk}_1\} \not\subseteq \mathsf{HUL}$ or $\mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}_0][\mathsf{item}] \neq \bot$
        or $\mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}_1][\mathsf{item}] \neq \bot$ then return $0$;
    $(\mathsf{ssk}, (\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}}_{t+1})) \leftarrow \left\langle \begin{matrix} \mathsf{Join}(\mathsf{upk}_b, \mathsf{usk}_b, \mathsf{ipk}, \mathsf{item}), \\ \mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{upk}_b, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}) \end{matrix} \right\rangle;$
    $d \leftarrow \mathcal{A}_2 \left\{ \begin{matrix} \mathsf{SndToKI}, \mathsf{OSign}^*_{\mathsf{upk}_0,\mathsf{upk}_1}, \mathsf{OTrace}, \\ \mathsf{RevUser}^*, \mathsf{PrivChal}_{b,\mathsf{upk}_0,\mathsf{upk}_1} \end{matrix} \right\} \left(\mathsf{st}, \mathsf{info}^{\mathsf{ipk}}_{t+1}\right);$
    if $|\mathsf{CL}| \neq 1$ then return $0$;
    return $d$;
```

Fig. 8: The experiment for defining BHS-PP security.

$\mathsf{KIgen}'(\mathsf{pp})$: It first runs $(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathbb{I}^{\mathsf{ipk}}) \leftarrow \mathsf{KIgen}(\mathsf{pp})$. Then, it generates a dummy user $(\mathsf{upk}', \mathsf{usk}') \leftarrow \mathsf{UKgen}(1^n)$. Finally, it outputs $(\mathsf{ipk}', \mathsf{isk}', \mathsf{reg}^{\mathsf{ipk}'}_{\mathsf{ki}}, \mathbb{I}^{\mathsf{ipk}}) \leftarrow ((\mathsf{ipk}, \mathsf{upk}'), (\mathsf{isk}, \mathsf{usk}'), \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}, \mathbb{I}^{\mathsf{ipk}})$.

$\langle \mathsf{Join}'(\mathsf{upk}, \mathsf{usk}, \mathsf{ipk}', \mathsf{item}), \mathsf{Issue}'(\mathsf{ipk}', \mathsf{isk}', \mathsf{reg}^{\mathsf{ipk}'}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}'}_t, \mathsf{upk}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}) \rangle$: It first checks if $\mathsf{info}^{\mathsf{ipk}'}_t$ contains $t = 0$. If so, it makes the user join the group by running $\langle \mathsf{Join}(\mathsf{upk}', \mathsf{usk}', \mathsf{ipk}, \mathsf{item}), \mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}'}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}'}_0, \mathsf{upk}', \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}) \rangle$, where $(\mathsf{ipk}, \mathsf{upk}') \leftarrow \mathsf{ipk}'$ and $(\mathsf{isk}, \mathsf{usk}') \leftarrow \mathsf{isk}'$. Then, it runs $(\mathsf{ssk}, (\mathsf{reg}^{\mathsf{ipk}'}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}'}_2)) \leftarrow \langle \mathsf{Join}(\mathsf{upk}, \mathsf{usk}, \mathsf{ipk}, \mathsf{item}), \mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}^{\mathsf{ipk}'}_{\mathsf{ki}}, \mathsf{info}^{\mathsf{ipk}'}_1, \mathsf{upk}, \mathsf{item}, \mathbb{I}^{\mathsf{ipk}}) \rangle$. Otherwise, it runs as the same algorithm as $\langle \mathsf{Join}, \mathsf{Issue} \rangle$.

The other algorithms ($\mathsf{RepSetup}', \mathsf{TMgen}', \mathsf{UKgen}', \mathsf{RevokeUser}', \mathsf{Sign}', \mathsf{Verify}', \mathsf{Link}', \mathsf{Trace}', \mathsf{Judge}'$) are the same as ARS except that if the algorithm takes ipk' (resp., isk'), then it first parses $(\mathsf{ipk}, \mathsf{upk}') \leftarrow \mathsf{ipk}'$ (resp., $(\mathsf{isk}, \mathsf{usk}') \leftarrow \mathsf{isk}'$) and runs the corresponding algorithm of ARS that takes as input ipk (resp., isk) instead of ipk' (resp., isk').

C.3 Proof of Theorem 4.1

Proof. Let us fix a PPT adversary $\mathcal{A}$ attacking the BHS join privacy of ARS' and the value of the security parameter n. The attack game used to define the BHS join privacy is defined in Figure 8. Let Game_0 be the original attack game, T_0 be the event that $\mathcal{A}'$ outputs 1 in Game_0. Our overall strategy for the proof is as follows. We shall define a sequence $\mathsf{Game}_0, \ldots, \mathsf{Game}_5$ of modified attack games. Each of the games $\mathsf{Game}_0, \ldots, \mathsf{Game}_5$ operates on the same underlying probability space. For any $1 \leq i \leq 5$, we let T_i be the event that $\mathcal{A}'$ outputs 1 in Game_i. Our strategy is to show that for $0 \leq i \leq 4$, the quantity $|\Pr[T_i] - \Pr[T_{i+1}]|$ is negligible.

The sequence of games $\mathsf{Game}_0, \ldots, \mathsf{Game}_5$ is as follows.

Game_0 : This is the same game as $\mathsf{Exp}^{\mathsf{bhs\text{-}pp\text{-}0}}_{\mathsf{ARS}',\mathcal{A}'}(n)$ in Figure 8, i.e., $\Pr[\mathsf{Exp}^{\mathsf{bhs\text{-}pp\text{-}0}}_{\mathsf{ARS}',\mathcal{A}'}(n) = 1] = \Pr[T_0]$.

Game_1 : In this game, it aborts if AddU returns $\perp$ when $\mathcal{A}'$ queries AddU. Other operations are the same as Game_0.

Game_2 : In this game, it returns Σ' for each query to $\mathsf{PrivChal}_{0,\mathsf{upk}_0,\mathsf{upk}_1}$ from $\mathcal{A}'$, where $\Sigma' \leftarrow \mathsf{Sign}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{ssk}', \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{item}, \mathsf{M})$. Other operations are the same as Game_1.

Game_3 : In this game, it runs the $\langle \mathsf{Join}', \mathsf{Issue}' \rangle$ protocol between upk_1 and KI instead of the protocol between upk_0 and KI. Other operations are the same as Game_2.

Game_4 : In this game, it returns Σ_1 for each query to $\mathsf{PrivChal}_{0,\mathsf{upk}_0,\mathsf{upk}_1}$ from $\mathcal{A}'$, where $\Sigma_1 \leftarrow \mathsf{Sign}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{ssk}_1, \mathsf{info}^{\mathsf{ipk}}_t, \mathsf{item}, \mathsf{M})$. We note that ssk_1 is the output of Join' when it runs $\langle \mathsf{Join}', \mathsf{Issue}' \rangle$ protocol between upk_1 and KI. Other operations are the same as Game_3.

Game_5 : In this game, it continues even if AddU returns $\perp$ when $\mathcal{A}'$ queries AddU. Other operations are the same as Game_4. Now, this is the same game as $\mathsf{Exp}^{\mathsf{bhs\text{-}pp\text{-}1}}_{\mathsf{ARS}',\mathcal{A}'}(n)$ in Figure 8, i.e, $\Pr\left[\mathsf{Exp}^{\mathsf{bhs\text{-}pp\text{-}1}}_{\mathsf{ARS}',\mathcal{A}'}(n) = 1\right] = \Pr[T_5]$.

Let us analyze the success probabilities. Firstly, we can ignore the probability that AddU' returns $\perp$ since the entropy of the output space of the UKgen algorithm is sufficiently large w.l.o.g.[13] Therefore,

$$|\Pr[T_0] - \Pr[T_1]| = \mathsf{negl}(n), \tag{1}$$
$$|\Pr[T_4] - \Pr[T_5]| = \mathsf{negl}(n). \tag{2}$$

Secondly, any difference between these games Game_1 and Game_2 yields a PPT algorithm that distinguishes two signatures generated by the honest user and the dummy user, respectively. More precisely, we have the following:

Lemma C.1. *There exists a PPT algorithm $\mathcal{A}$ such that*

$$|\Pr[T_1] - \Pr[T_2]| \leq \mathsf{Adv}^{\mathsf{anon}}_{\mathsf{ARS},\mathcal{A}}(n). \tag{3}$$

Detailed proof of this lemma is provided in the supplemental material C.4. We can say the same thing about the difference between these two games Game_3 and Game_4. More precisely, we have the following:

Lemma C.2. *There exists a PPT algorithm $\mathcal{A}$ such that*

$$|\Pr[T_3] - \Pr[T_4]| \leq \mathsf{Adv}^{\mathsf{anon}}_{\mathsf{ARS},\mathcal{A}}(n). \tag{4}$$

The proof of this lemma is almost identical to the proof of Lemma C.1 .

Finally, we analyze the difference between the success probabilities of the two games Game_2 and Game_3. Since ARS satisfies the purchase privacy, we have the following facts:

[13] We assume that each user has its own unique key. Therefore, if the entropy is small, then the secret key corresponding to upk possibly leak.

- $\mathsf{info}_{t+1}^{\mathsf{ipk}}$ generated by Issue_1 does not depend on upk_0 or upk_1, i.e., the challenge bit of the game. In addition, $\mathcal{A}'$ cannot get any information about messages ρ_1, ρ_2, updated $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$, and the output of the Join algorithm ssk.
- no $\mathsf{info}_{t+1}^{\mathsf{ipk}}$ obtained by querying to the $\mathsf{RevUser}$ oracle depends on upk_0 or upk_1 since $\mathcal{A}'$ is not allowed to revoke them.

Therefore, two games Game_2 and Game_3 proceed identically until the following event F occurs: $\mathcal{A}_2'$ receives a signature of the user upk_0 or upk_1 chosen by $\mathcal{A}_1'$. Thus, we have $|\Pr[T_2] - \Pr[T_3]| \le \Pr[F]$. In addition, the event F never occurs since the signing oracle $\mathsf{OSign}^*_{\mathsf{upk}_0,\mathsf{upk}_1}$ returns $\perp$ and the challenge oracle $\mathsf{PrivChal}_{b,\mathsf{upk}_0,\mathsf{upk}_1}$ returns a signature of the dummy user. Therefore, we have

$$\Pr[T_2] = \Pr[T_3]. \tag{5}$$

Theorem 4.1 now follows immediately from (1) - (5).

C.4 Proof of Lemma C.1

Proof. We assume that there exists a PPT adversary $\mathcal{A}' = (\mathcal{A}_1', \mathcal{A}_2')$ that makes the probability $|\Pr[T_1] - \Pr[T_2]|$ non-negligible in the security parameter. Then, we can construct another PPT adversary $\mathcal{A}$ attacking the anonymity of ARS with success probability $|\Pr[T_1] - \Pr[T_2]|$, which implies this lemma. The description of $\mathcal{A}$ is as follows.

$\mathcal{A}$, given $(\mathsf{pp}, \mathsf{tpk})$, computes $(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}, \mathbb{I}^{\mathsf{ipk}}) \leftarrow \mathsf{KIgen}(\mathsf{pp})$, generates a dummy user by querying $\mathsf{AddU}()$, receives the dummy user's public key upk', and sets $\mathsf{ipk}' \leftarrow (\mathsf{ipk}, \mathsf{upk}')$, $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}'} \leftarrow \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$, $\mathbb{I}^{\mathsf{ipk}'} \leftarrow \mathbb{I}^{\mathsf{ipk}}$, $\mathsf{HKIL}' := \{\mathsf{ipk}'\}$, and $\mathsf{HUL}', \mathsf{HSSKL}', \mathsf{CL}', \mathsf{SL}', \mathsf{IsActive}' = \emptyset$. Then, $\mathcal{A}$ runs $\mathcal{A}_1'(\mathsf{pp}, \mathsf{ipk}', \mathsf{tpk}, \mathbb{I}^{\mathsf{ipk}'})$. $\mathcal{A}$ responds for each of queries from $\mathcal{A}_1'$ as follows:

- For each query to $\mathsf{AddU}()$, $\mathcal{A}$ similarly queries AddU. If AddU outputs $\perp$ to $\mathcal{A}$, then it aborts. Otherwise, $\mathcal{A}$ returns the received upk to $\mathcal{A}_1'$ and adds $\mathsf{HUL}' \leftarrow \mathsf{HUL}' \cup \{\mathsf{upk}\}$.
- For each query to $\mathsf{SndToKI}(\mathsf{ipk}', \mathsf{upk}, \mathsf{item})$, $\mathcal{A}$ returns $\perp$ if $\mathsf{ipk}' \notin \mathsf{HKIL}'$. Otherwise, it runs $\langle \mathcal{A}_1', \mathsf{Issue}'(\mathsf{ipk}', \mathsf{isk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}'}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}'}, \mathsf{upk}, \mathsf{item})\rangle$. Then, $\mathcal{A}$ updates $\mathsf{IsActive}'[\mathsf{ipk}'][\mathsf{item}][\mathsf{upk}] = [0, \infty]$ and returns $\mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}+1}^{\mathsf{ipk}'}$ included in the output of the Issue' algorithm.
- For each query to $\mathsf{SndToKIU}(\mathsf{ipk}', \mathsf{upk}, \mathsf{item})$, $\mathcal{A}$ returns $\perp$ if $\mathsf{ipk}' \notin \mathsf{HKIL}'$ or $\mathsf{upk} \notin \mathsf{HUL}'$. Otherwise, in case $t_{\mathsf{cur}}^{\mathsf{ipk}} = 0$, $\mathcal{A}$ first queries $\mathsf{SndToU}(\mathsf{ipk}, \mathsf{upk}', \mathsf{item})$, where $(\mathsf{ipk}, \mathsf{upk}') \leftarrow \mathsf{ipk}'$. Then, $\mathcal{A}$ queries $\mathsf{SndToU}(\mathsf{ipk}, \mathsf{upk}, \mathsf{item})$, where $\mathcal{A}$ plays the role of the honest key issuer, i.e., $\mathsf{Issue}(\mathsf{ipk}, \mathsf{isk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}'}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}}^{\mathsf{ipk}'}, \mathsf{upk}, \mathsf{item})$, where isk is the secret key corresponding to ipk. Then, $\mathcal{A}$ updates $\mathsf{IsActive}'[\mathsf{ipk}']$ $[\mathsf{item}][\mathsf{upk}] = [0, \infty]$, and returns $\mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}}+1}^{\mathsf{ipk}'}$ included in the output of the Issue algorithm.

- For each query to $\mathsf{RevUser}(R)$, $\mathcal{A}$ computes $(\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}'}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}'}+1}^{\mathsf{ipk}'}) \leftarrow \mathsf{RevokeUser}$ $(\mathsf{isk}, R, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}'}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}'}}^{\mathsf{ipk}'})$, where $(\mathsf{ipk}, -) \leftarrow \mathsf{ipk}'$ and isk is the secret key corresponding to ipk. Then, it updates $\mathsf{IsActive}'[\mathsf{ipk}'][\mathsf{item}][\mathsf{upk}] = [0, t_{\mathsf{cur}}^{\mathsf{ipk}}]$ for all $\mathsf{upk} \in R$ and $\mathsf{item} \in \mathbb{I}^{\mathsf{ipk}}$ such that $\mathsf{IsActive}'[\mathsf{ipk}'][\mathsf{item}][\mathsf{upk}] = [0, \infty]$.
- For each query to $\mathsf{OSign}(\mathsf{ipk}', \mathsf{info}_t^{\mathsf{ipk}'}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{A}$ parses $(\mathsf{ipk}, -) \leftarrow \mathsf{ipk}'$, queries $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}_t^{\mathsf{ipk}'}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, and receives Σ. Then, $\mathcal{A}$ updates $\mathsf{SL}' \leftarrow \mathsf{SL}' \cup \{(t, \mathsf{ipk}', \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma)\}$ and returns Σ to $\mathcal{A}_1'$, where t is included in $\mathsf{info}_t^{\mathsf{ipk}'}$.
- For each query to $\mathsf{OTrace}(\mathsf{ipk}', \mathsf{item}, \mathsf{info}, \mathsf{M}, \Sigma)$, $\mathcal{A}$ parses $(\mathsf{ipk}, -) \leftarrow \mathsf{ipk}'$, queries $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{item}, \mathsf{info}, \mathsf{M}, \Sigma)$, and receives $(\mathsf{upk}, \Pi_{\mathsf{Trace}})$. Then, $\mathcal{A}$ returns $(\mathsf{upk}, \Pi_{\mathsf{Trace}})$ to $\mathcal{A}_1'$.

When $\mathcal{A}_1'$ outputs $(\mathsf{st}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item})$ and terminates, $\mathcal{A}$ outputs 0 and terminates if $\{\mathsf{upk}_0, \mathsf{upk}_1\} \nsubseteq \mathsf{HUL}'$ or $\mathsf{HSSKL}'[\mathsf{ipk}'][\mathsf{upk}_0][\mathsf{item}] \neq \bot$ or $\mathsf{HSSKL}'[\mathsf{ipk}']$ $[\mathsf{upk}_1][\mathsf{item}] \neq \bot$. Otherwise, $\mathcal{A}$ queries $\mathsf{SndToU}(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{item})$ and runs $\mathcal{A}_2'(\mathsf{st}, \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}'}}^{\mathsf{ipk}'})$. $\mathcal{A}$ responds for each of queries from $\mathcal{A}_2'$ as follows:

- For each of queries to $\mathsf{AddU}, \mathsf{SndToKI}, \mathsf{SndToKIU}, \mathsf{RevUser}$, $\mathcal{A}$ similarly makes each response to $\mathcal{A}_2'$ the same way it does for each of queries from $\mathcal{A}_1'$.
- For each query to $\mathsf{PrivChal}_{b,\mathsf{upk}_0,\mathsf{upk}_1}(\mathsf{ipk}', \mathsf{info}_{t_{\mathsf{cur}}^{\mathsf{ipk}'}}^{\mathsf{ipk}'}, \mathsf{item}, \mathsf{M})$, $\mathcal{A}$ queries $\mathsf{AnonChal}_b$ $(\mathsf{ipk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{upk}_0, \mathsf{upk}', \mathsf{item}, \mathsf{M})$. If $\mathcal{A}$ receives $\bot$, then it returns $\bot$ to $\mathcal{A}_2'$. Otherwise, $\mathcal{A}$ receives Σ, so it updates $\mathsf{CL}' \leftarrow \mathsf{CL}' \cup \{(\mathsf{ipk}', \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma)\}$ and returns Σ to $\mathcal{A}_2'$.
- For each query to $\mathsf{OSign}_{\mathsf{upk}_0,\mathsf{upk}_1}^{*}(\mathsf{ipk}', \mathsf{info}_t^{\mathsf{ipk}'}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{A}$ outputs 0 if $\mathsf{upk} = \mathsf{upk}_0$ or $\mathsf{upk} = \mathsf{upk}_1$. Otherwise, $\mathcal{A}$ queries $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}_t^{\mathsf{ipk}'}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$ and receives Σ, where $(\mathsf{ipk}, -) \leftarrow \mathsf{ipk}'$. Then, it returns Σ to $\mathcal{A}_2'$ and updates $\mathsf{SL}' \leftarrow \mathsf{SL}' \cup \{(t, \mathsf{ipk}', \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma)\}$, where t is included in $\mathsf{info}_t^{\mathsf{ipk}'}$.
- For each query to $\mathsf{OTrace}(\mathsf{ipk}', \mathsf{item}, \mathsf{info}, \mathsf{M}, \Sigma)$, $\mathcal{A}$ returns $\bot$ if $\exists(\mathsf{ipk}', -, -, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{CL}'$. Otherwise, it parses $(\mathsf{ipk}, -) \leftarrow \mathsf{ipk}'$, queries $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{item}, \mathsf{info}, \mathsf{M}, \Sigma)$, and receives $(\mathsf{upk}, \Pi_{\mathsf{Trace}})$. Then, $\mathcal{A}$ returns $(\mathsf{upk}, \Pi_{\mathsf{Trace}})$ to $\mathcal{A}_1'$.

When $\mathcal{A}_2'$ outputs d and terminates, $\mathcal{A}$ outputs d and terminates.

The above completes the description of $\mathcal{A}$. Note that $\mathcal{A}$ perfectly simulates Game_1 (resp., Game_2) in case the challenge bit $b = 0$ (resp., $b = 1$) for $\mathcal{A}'$ since $\mathcal{A}'$ receives a signature generated by upk_0 (resp., upk') from the query to the $\mathsf{PrivChal}_{0,\mathsf{upk}_0,\mathsf{upk}_1}$ oracle. Therefore, we have $|\Pr[T_1] - \Pr[T_2]| \leq \mathsf{Adv}_{\mathsf{ARS},\mathcal{A}}^{\mathsf{anon}}(n)$.

D Security Proofs of Our Generic Construction

D.1 Proof of Theorem 5.2

Proof. [14] Let us fix a PPT adversary $\mathcal{A}$ attacking anonymity of the ARS and the value of the security parameter n. The attack game used to define anonymity

[14] We would like to thank [19] since we referred it well to complete this proof.

is defined in Figure 2. Let Game_0^b be the original attack game, where b is the challenge bit, and let T_0^b be the event that $\mathcal{A}$ outputs 1 in the Game_0^b. Our overall strategy for the proof is as follows. We shall define a sequence $\mathsf{Game}_0^b, ..., \mathsf{Game}_3^b$ of modified attack games. Each of the games $\mathsf{Game}_0^b, ..., \mathsf{Game}_3^b$ operates on the same underlying probability space. For any $1 \le i \le 3$, we let T_i^b be the event that $\mathcal{A}$ outputs 1 in the Game_i^b. Our strategy is to show that for $0 \le i \le 2$ and $b \in \{0,1\}$, the quantity $|\Pr[T_i^b] - \Pr[T_{i+1}^b]|$ is negligible and the probability $|\Pr[T_3^0] - \Pr[T_3^1]|$ is negligible.

So that the overall structure of the proof is more transparent, we shall defer the proofs of all lemmas to the end of the proof of the theorem.

Game_1^b. We now modify Game_0^b to obtain a new game Game_1^b. These two games are identical, except for a small modification to the $\mathsf{RepSetup}$ algorithm and the OTrace oracle. Instead of using the original Trace algorithm to compute a tracing result and its proof, we use modified algorithms, in which some steps are replaced by as follows:

- In the $\mathsf{RepSetup}$ algorithm, $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$ is replaced by $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}_0^2(1^n)$.
- In the Trace algorithm, $\Pi_{\mathsf{Trace}} \leftarrow \mathsf{NIZK.Prove}_2(\mathsf{crs}_2, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$ is replaced by $\mathsf{Sim}_1^2(\mathsf{crs}_2, \mathsf{td}_2, \mathsf{lbl}, \mathsf{X})$, where $\mathsf{lbl} \leftarrow \epsilon$, $\mathsf{X} \leftarrow \langle \mathsf{ek}_1, c_1, \mathsf{upk}\rangle$, and $\mathsf{W} \leftarrow \mathsf{tsk}$.

Any difference between these two games yields a PPT algorithm that distinguishes the real proof from the simulated proof. More precisely, we have:

Lemma D.1. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$|\Pr[T_1^b] - \Pr[T_0^b]| \le \mathsf{Adv}_{\Pi_2,\mathcal{B}}^{\mathsf{zk}}(n). \tag{6}$$

Game_2^b. We modify Game_1^b to obtain a new game Game_2^b. These two games are identical, except for a small modification to the $\mathsf{RepSetup}$ algorithm and two oracles OSign and $\mathsf{AnonChal}_b$. Instead of using the original Sign algorithm to compute signatures, we use modified algorithms, in which some steps are replaced by as follows:

- In the $\mathsf{RepSetup}$ algorithm, $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$ is replaced by $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{Sim}_0^1(1^n)$.
- In the Sign algorithm, $\pi \leftarrow \mathsf{NIZK.Prove}_1(\mathsf{crs}_1, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$ is replaced by $\mathsf{Sim}_1^1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{lbl}, \mathsf{X})$, where $\mathsf{lbl} \leftarrow \mathsf{M}$, $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle$, and $\mathsf{W} \leftarrow \langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1.r_2\rangle$.

Similar to the above, any difference between these two games yields a PPT algorithm that distinguishes the real proof from the simulated proof. More precisely, we have:

Lemma D.2. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$|\Pr[T_2^b] - \Pr[T_1^b]| \le \mathsf{Adv}_{\Pi_1,\mathcal{B}}^{\mathsf{zk}}(n). \tag{7}$$

Game_3^b. In the following few steps, we modify the $\mathsf{AnonChal}_b$ oracle in Game_2^b to obtain a new game Game_3^b. Instead of using the original Sign algorithm to compute the target signature, we use the modified Sign algorithm described as follows:

- $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}_b; r_1)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}_b; r_2)$ are replaced by $\widetilde{c_1} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, 0^{|\mathsf{upk}_b|}; r_1)$ and $\widetilde{c_2} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, 0^{|\mathsf{upk}_b|}; r_2)$, respectively.

Any difference between these two games yields a PPT algorithm that distinguishes a ciphertext of upk_b from a ciphertext of all-zero string of length equal to that of upk_b. More precisely, we have:

Lemma D.3. *There exists a PPT algorithm $\mathcal{B}_1, \mathcal{B}_2$ such that*

$$|\Pr[T_3^b] - \Pr[T_2^b]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}_1}(n) + 2 \cdot \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}_2}(n). \tag{8}$$

Finally, we show that any difference between two games T_3^0 and T_3^1 yields a PPT algorithm that distinguishes a tag computed using upk_0 from one computed using upk_1. More precisely:

Lemma D.4. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$|\Pr[T_3^0] - \Pr[T_3^1]| \leq \mathsf{Adv}^{\mathsf{tag\text{-}ind}}_{\mathsf{LIT},\mathcal{B}}(n). \tag{9}$$

Theorem 5.2 now follows immediately from (6)-(9).

Proofs of Lemmas.

To complete the proof of Theorem 5.2, we now present the proofs of Lemmata D.1 to D.4. In all proofs except the last one, we fix the challenge bit b.

Proof of Lemma D.1.

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $|\Pr[T_0^b] - \Pr[T_1^b]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the zero-knowledge property of Π_2 with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ initially receives crs_2, computes $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$, and sets $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$. It also computes $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TMgen}(\mathsf{pp})$ and sets $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk})$. It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to AddU, SndToU, $\mathsf{AnonChal}_b$, OSign, and $\mathsf{RevUser}$, $\mathcal{B}$ operates the same as defined in Section B.1.
- For each query to $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma)$, $\mathcal{B}$ outputs $\perp$ if there exists $(\mathsf{ipk}, -, -, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{CL}$ or $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma) = 0$. Otherwise, it parses $((c_1, c_2), \tau, \pi) \leftarrow \Sigma$ and $(\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}) \leftarrow \mathsf{tsk}$, and computes $\mathsf{upk} \leftarrow \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$. If $\mathsf{upk} = \perp$, then it outputs $\perp$. Otherwise, it sets $\mathsf{lbl} \leftarrow \epsilon$, $\mathsf{X} \leftarrow \langle \mathsf{ek}_1, c_1, \mathsf{upk} \rangle$, and $\mathsf{W} \leftarrow \mathsf{tsk}$. Then, $\mathcal{B}$ queries $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ to the challenge oracle, and receives Π_{Trace}. Finally, it returns $(\mathsf{upk}, \Pi_{\mathsf{Trace}})$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}$ outputs 0 if $|\mathsf{CL}| \neq 1$ or there exists $(\mathsf{ipk}, \mathsf{upk}_i, \mathsf{item}, -, -) \in \mathsf{SL}$ for $i = 0$ or 1, where $(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma_b) \leftarrow \mathsf{CL}$. Otherwise, $\mathcal{B}$ outputs b' and terminates.

The above completes the description of $\mathcal{B}$. If crs_2 is generated by the $\mathsf{NIZK.Setup}_2$ (resp., Sim_0^1) algorithm and $\mathcal{B}$ accesses the $\mathsf{NIZK.Prove}_2$ (resp., Sim_1^1) oracle, then $\mathcal{B}$ perfectly simulates Game_0^b (resp., Game_1^b) for $\mathcal{A}$. Therefore, we have $|\Pr[T_0^b] - \Pr[T_1^b]| = \mathsf{Adv}^{\mathsf{zk}}_{\Pi_2,\mathcal{B}}(n)$.

Proof of Lemma D.2.

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $|\Pr[T_2^b] - \Pr[T_1^b]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the zero-knowledge property of Π_1 with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ initially receives crs_1, computes $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}_0^2(1^n)$, and sets $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$. It also computes $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TMgen}(\mathsf{pp})$ and sets $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk})$. It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to AddU, SndToU, and $\mathsf{RevUser}$, $\mathcal{B}$ operates the same as defined in Section B.1.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}$ makes a response as follows:
  Check: Return $\perp$ if $\mathsf{ssk} = \perp$ where $\mathsf{ssk} \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}]$. Otherwise, continue.
  Compute $C = (c_1, c_2)$: Choose r_1 and r_2 uniform randomly, then compute $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2)$.
  Compute τ: Compute $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}, \langle \mathsf{ipk}, \mathsf{item} \rangle)$, where $(\mathsf{upk}, \mathsf{usk}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}}, \theta) \leftarrow \mathsf{ssk}$.
  Compute π_{acc}: Compute $\pi_{\mathsf{acc}} \leftarrow \mathsf{ACC.Prove}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{aux}_{\mathsf{acc}}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}})$, where $\mathsf{pp}_{\mathsf{acc}} \leftarrow \mathsf{ipk}$ and $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$.
  Query: Set $\mathsf{lbl} \leftarrow \mathsf{M}$, $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$, and $\mathsf{W} \leftarrow \langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle$. Then, send $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ to the challenger for zero-knowledge of Π_1, and receive π.
  Finally, it returns (C, τ, π) to $\mathcal{A}$.
- For each query to $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{item}, \mathsf{M}, \Sigma)$, $\mathcal{B}$ operates the same as defined in Section B.1 except that it returns a simulated proof instead of a real proof.
- For each query to $\mathsf{AnonChal}_b(\mathsf{ipk}, \mathsf{info}_t^{\mathsf{ipk}}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M})$, $\mathcal{B}$ makes a response as follows:
  Check: Return $\perp$ if $\mathsf{ssk}_i = \perp$ where $\mathsf{ssk}_i \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}_i][\mathsf{item}]$ for $i = 0$ or 1. Otherwise, continue.
  Compute $C = (c_1, c_2)$: Choose r_1 and r_2 uniform randomly, compute $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}_b; r_1)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}_b; r_2)$.
  Compute τ_b: Compute $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}_b, \langle \mathsf{ipk}, \mathsf{item} \rangle)$, where $(\mathsf{upk}_b, \mathsf{usk}_b, \mathsf{id}_b, \mathsf{wit}_{\mathsf{id}_b}, \theta_b) \leftarrow \mathsf{ssk}_b$.
  Compute $\pi_{\mathsf{acc},b}$: Compute $\pi_{\mathsf{acc},b} \leftarrow \mathsf{ACC.Prove}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{aux}_{\mathsf{acc}}, \mathsf{id}_b, \mathsf{wit}_{\mathsf{id}_b})$, where $\mathsf{pp}_{\mathsf{acc}} \leftarrow \mathsf{ipk}$ and $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}_t^{\mathsf{ipk}}$.

Query: Set $\mathsf{lbl} \leftarrow \mathsf{M}$, $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau_b \rangle$, and $\mathsf{W} \leftarrow \langle \mathsf{upk}_b, \mathsf{usk}_b, \mathsf{id}_b, \pi_{\mathsf{acc},b}, \theta_b, r_1, r_2 \rangle$. Then, send $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ to the challenger for zero-knowledge of Π_1, and receive π.

Finally, it returns (C, τ, π) to $\mathcal{A}$.

When $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}$ outputs 0 and terminates if $|\mathsf{CL}| \neq 1$ or there exists $(\mathsf{ipk}, \mathsf{upk}_i, \mathsf{item}, -, -) \in \mathsf{SL}$ for $i = 0$ or 1, where $(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma_b) \leftarrow \mathsf{CL}$. Otherwise, $\mathcal{B}$ outputs b' and terminates.

The above completes the description of $\mathcal{B}$. If crs_1 is generated by the $\mathsf{NIZK.Setup}_1$ (resp., Sim_0^1) algorithm and $\mathcal{B}$ accesses the $\mathsf{NIZK.Prove}_1$ (resp., Sim_1^1) oracle, then $\mathcal{B}$ perfectly simulates Game_1^b (resp., Game_2^b) for $\mathcal{A}$. Therefore, we have $|\Pr[T_1^b] - \Pr[T_2^b]| = \mathsf{Adv}^{\mathsf{zk}}_{\Pi_1,\mathcal{B}}(n)$.

Proof of Lemma D.3.

Proof. We use the Naor-Yung paradigm [33] to prove this lemma. In particular, we consider the following three different games, and informal descriptions are as follows:

- $\mathsf{Game}^b_{2\text{-}1}$**:** Like Game_2^b except that c_2 is computed as an encryption of $0^{|\mathsf{upk}_b|}$.
- $\mathsf{Game}^b_{2\text{-}2}$**:** Like $\mathsf{Game}^b_{2\text{-}1}$ except decrypt c_2 using dk_2 instead of c_1 using dk_1.
- $\mathsf{Game}^b_{2\text{-}3}$**:** Like $\mathsf{Game}^b_{2\text{-}2}$ except that c_1 is computed as an encryption of $0^{|\mathsf{upk}_b|}$.

Let $T^b_{2\text{-}1}, T^b_{2\text{-}2}, T^b_{2\text{-}3}$ denote the event that $\mathcal{A}$ outputs 1 in the $\mathsf{Game}^b_{2\text{-}1}, \mathsf{Game}^b_{2\text{-}2}, \mathsf{Game}^b_{2\text{-}3}$, respectively. Note that Game_3^b is like $\mathsf{Game}^b_{2\text{-}3}$ except use dk_1 to decrypt instead of dk_2. We show at each step, with the construction of each new game, that the ability of the adversary to distinguish between the two games is negligible.

Firstly, we show there exists a PPT algorithm $\mathcal{B}_1$ such that $|\Pr[T_3^b] - \Pr[T^b_{3\text{-}1}]| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}_1}$. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $|\Pr[T_3^b] - \Pr[T^b_{3\text{-}1}]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}_1$ against the IND-CPA security of PKE. The description of $\mathcal{B}_1$ is as follows.

$\mathcal{B}_1$ initially receives ek_2, computes $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n)$, $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{Sim}_0^1(1^n)$ and $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}_0^2(1^n)$, and sets $\mathsf{tpk} := (\mathsf{ek}_1, \mathsf{ek}_2)$, $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}_1$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk})$. It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to AddU and SndToU, $\mathcal{B}_1$ operates the same as defined in Section B.1.
- For each of queries to OSign and OTrace, $\mathcal{B}_1$ operates the same as defined in Section B.1 except that it contains a simulated proof instead of a real proof.
- For each query to $\mathsf{AnonChal}_b(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M})$, $\mathcal{B}_1$ makes a response as follows:

  Check: Return $\perp$ if $\mathsf{ssk}_i = \perp$ where $\mathsf{ssk}_i \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}_i][\mathsf{item}]$ for $i = 0$ or 1. Otherwise, continue.

Query: Compute $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}_b)$ and set $m_0 \leftarrow \mathsf{upk}_b$ and $m_1 \leftarrow 0^{|\mathsf{upk}_b|}$. Then, send (m_0, m_1) to the challenger for the IND-CPA security of PKE, receive c^*, and set $C^* \leftarrow (c_1, c^*)$.

Compute τ_b: Compute $\tau_b \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}_b, \langle \mathsf{ipk}, \mathsf{item} \rangle)$, where $(\mathsf{upk}_b, \mathsf{usk}_b, \mathsf{wit}_b) \leftarrow \mathsf{ssk}_b$.

Compute $\widetilde{\pi}$: Compute $\widetilde{\pi} \leftarrow \mathsf{Sim}^1_1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{M}, \mathsf{X})$, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$ and $\mathsf{X} \leftarrow \langle C^*, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau_b \rangle$.

Finally, it returns $(C^*, \tau, \widetilde{\pi})$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}_1$ outputs 0 and terminates if $|\mathsf{CL}| \neq 1$ or there exists $(\mathsf{ipk}, \mathsf{upk}_i, \mathsf{item}, -, -) \in \mathsf{SL}$ for $i = 0$ or 1, where $(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma_b) \leftarrow \mathsf{CL}$. Otherwise, $\mathcal{B}_1$ outputs b' and terminates.

The above completes the description of $\mathcal{B}_1$. Let d be the challenge bit for $\mathcal{B}_1$ in the IND-CPA security game. If $d = 0$, c^* is represented as $c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}_b)$. On the other hand, if $d = 1$, c^* is represented as $c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, 0^{|\mathsf{upk}_b|})$. We note that in both games, $\mathcal{B}_1$ need not to know dk_2 for responding to a query to the OTrace oracle. Thus, $\mathcal{B}_1$ perfectly simulates Game^b_2 if $d = 0$ and $\mathsf{Game}^b_{2\text{-}1}$ if $d = 1$, respectively. Therefore, we have $|\Pr[T^b_2] - \Pr[T^b_{2\text{-}1}]| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}_1}(n)$.

Secondly, we show there exists a PPT algorithm $\mathcal{B}_2$ such that $|\Pr[T^b_{2\text{-}1}] - \Pr[T^b_{2\text{-}2}]| \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}_2}$. Let fake be the event in the $\mathsf{Game}^b_{2\text{-}2}$ that the adversary queries $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{item}, \mathsf{info}, \mathsf{M}, \Sigma)$ such that $\mathsf{PKE.Dec}(\mathsf{dk}_1, c_1) \neq \mathsf{PKE.Dec}(\mathsf{dk}_2, c_2)$ but $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 1$, where $((c_1, c_2), -, -) \leftarrow \Sigma$. Since the adversary's views in the $\mathsf{Game}_{2\text{-}1}$ and the $\mathsf{Game}_{2\text{-}2}$ are identical until the event fake occurs, we have $|\Pr[T^b_{2\text{-}1}] - \Pr[T^b_{2\text{-}2}]| \leq \Pr[\mathsf{fake}]$. Thus, it is enough to show $\Pr[\mathsf{fake}] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}_2}$.

We first modify the game slightly as follows. In the $\mathsf{RepSetup}$ algorithm, we compute $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}^1_0(1^n)$ instead of $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$. Note that its affects nothing about the probability $\mathcal{A}$ wins.

Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[\mathsf{fake}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}_2$ against the simulation-extractability of Π_1. The description of $\mathcal{B}_2$ is as follows.

$\mathcal{B}_2$ initially receives crs_1, computes $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n)$, $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$ and $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}^2_0(1^n)$, and sets $\mathsf{tpk} := (\mathsf{ek}_1, \mathsf{ek}_2)$, $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}_2$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk})$. It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to AddU, SndToU, and $\mathsf{AnonChal}_b$, $\mathcal{B}_2$ operates the same way as $\mathcal{B}_1$ does in the above.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}_2$ computes $(C, \tau, \pi_{\mathsf{acc}})$ in the same as defined in Section B.1, queries $(\mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle)$ to the Sim^1_1 oracle, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma_{\mathsf{acc}}) \leftarrow \mathsf{info}$, and receives a proof π. Finally, it returns (C, τ, π) to $\mathcal{A}$.
- For each query to $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$, $\mathcal{B}_2$ makes a response as follows. If the event fake occurs, then $\mathcal{B}_2$ outputs $(\mathsf{M}, (C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau), \widetilde{\pi})$, where $(C, \tau, \widetilde{\pi}) \leftarrow \Sigma$. Otherwise, $\mathcal{B}_2$ returns the same as defined in Section B.1 except that it contains a simulated proof that $\mathcal{B}_2$ can obtain by querying the Sim^1_1 oracle, instead of a real proof.

When $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}_2$ outputs 0 and terminates.

The above completes the description of $\mathcal{B}_2$. Let $\mathsf{W} \leftarrow \mathsf{SimExt}_1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$, where $\mathsf{X} \leftarrow (C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau)$. We note that the event fake occurs, $(\mathsf{X}, \mathsf{W}) \notin \rho_1 \wedge \mathsf{NIZK.Verify}(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \widetilde{\pi}) = 1$ since we clearly have $\mathsf{X} \notin L_{\rho_1}$, where $L_{\rho_1} := \{\mathsf{X} \mid \exists \mathsf{W}; (\mathsf{X}, \mathsf{W}) \in \rho_1\}$. We need to show $(\mathsf{M}, \mathsf{X}, \widetilde{\pi}) \notin L_{\mathcal{S}}$, where $L_{\mathcal{S}}$ is the list of queries and responses to Sim_1. $\mathcal{A}$ possibly gets a simulated proof $\widetilde{\pi}$ via querying to the OSign or $\mathsf{AnonChal}_b$ oracle. However, in case $\mathcal{B}_2$ queries a statement X to the OSign oracle and it returns $\widetilde{\pi}$, we always have $\mathsf{X} \in L_{\rho_1}$, thus the event fake never occurs. In addition, $\widetilde{\pi}$ is never included in a response from a query to the $\mathsf{AnonChal}_b$ oracle since all responds from the $\mathsf{AnonChal}_b$ oracle are included in CL and the OTrace oracle returns $\perp$ if there exists $(\mathsf{ipk}, -, -, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{CL}$. Therefore, we have $\Pr[\mathsf{fake}] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1, \mathcal{B}_2}(n)$.

Thirdly, we show there exists a PPT algorithm $\mathcal{B}_3$ such that $|\Pr[T^b_{2\text{-}2}] - \Pr[T^b_{2\text{-}3}]| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE}, \mathcal{B}_3}$. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $|\Pr[T^b_{2\text{-}2}] - \Pr[T^b_{2\text{-}3}]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}_3$ against the IND-CPA security of PKE. The description of $\mathcal{B}_3$ is as follows.

$\mathcal{B}_3$ initially receives ek_1, computes $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$, $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{Sim}^1_0(1^n)$ and $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}^2_0(1^n)$, and sets $\mathsf{tpk} := (\mathsf{ek}_1, \mathsf{ek}_2), \mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}_3$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk})$. It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to AddU, SndToU, OSign, and OTrace, $\mathcal{B}_3$ operates the same way as $\mathcal{B}_2$ does in the above.
- For each query to $\mathsf{AnonChal}_b(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M})$, $\mathcal{B}_3$ makes a response as follows:
  Check: Return $\perp$ if $\mathsf{ssk}_i = \perp$ where $\mathsf{ssk}_i \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}_i][\mathsf{item}]$ for $i = 0$ or 1. Otherwise, continue.
  Query: Compute $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}_b)$ and set $m_0 \leftarrow \mathsf{upk}_b$ and $m_1 \leftarrow 0^{|\mathsf{upk}_b|}$. Then, send (m_0, m_1) to the challenger for the IND-CPA security of PKE, receive c^*, and set $C^* \leftarrow (c^*, c_2)$.
  Compute τ_b: Compute $\tau_b \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}_b, \langle \mathsf{ipk}, \mathsf{item} \rangle)$, where $(\mathsf{upk}_b, \mathsf{usk}_b, \mathsf{id}_b, \mathsf{wit}_{\mathsf{id}_b}, \theta_b) \leftarrow \mathsf{ssk}_b$.
  Compute $\widetilde{\pi}$: Compute $\widetilde{\pi} \leftarrow \mathsf{Sim}^1_1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{M}, \mathsf{X})$, where $\mathsf{X} \leftarrow \langle C^*, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma_{\mathsf{acc}}) \leftarrow \mathsf{info}$.
  Finally, it returns $(C^*, \tau, \widetilde{\pi})$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}_3$ outputs 0 and terminates if $|\mathsf{CL}| \neq 1$ or there exists $(\mathsf{ipk}, \mathsf{upk}_i, \mathsf{item}, -, -) \in \mathsf{SL}$ for $i = 0$ or 1, where $(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma_b) \leftarrow \mathsf{CL}$. Otherwise, $\mathcal{B}_3$ outputs b' and terminates.

The above completes the description of $\mathcal{B}_3$. Let d be the challenge bit for $\mathcal{B}_3$ in the IND-CPA security game. If $d = 0$, c^* is represented as $c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}_b)$. On the other hand, if $d = 1$, c^* is represented as $c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, 0^{|\mathsf{upk}_b|})$. We note that in both games, $\mathcal{B}_3$ need not to know dk_1 for responding to a query to the OTrace oracle. Thus, $\mathcal{B}_3$ perfectly simulates $\mathsf{Game}^b_{2\text{-}2}$ if $d = 0$ and $\mathsf{Game}^b_{2\text{-}3}$ if $d = 1$, respectively. Therefore, we have $|\Pr[T^b_{2\text{-}2}] - \Pr[T^b_{2\text{-}3}]| = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE}, \mathcal{B}_3}(n)$.

Finally, we show there exists a PPT algorithm $\mathcal{B}_4$ such that $|\Pr[T^b_{2\text{-}3}] - \Pr[T^b_3]| \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}_4}$. As in the proof of indistinguishability between $\mathsf{Game}^b_{2\text{-}1}$ and $\mathsf{Game}^b_{2\text{-}2}$, let fake' be the event in the Game^b_3 that the adversary queries $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{item}, \mathsf{info}, \mathsf{M}, \Sigma)$ such that $\mathsf{PKE.Dec}(\mathsf{dk}_1, c_1) \neq \mathsf{PKE.Dec}(\mathsf{dk}_2, c_2)$ but $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 1$, where $((c_1, c_2), -, -) \leftarrow \Sigma$. Since the adversary's views in the $\mathsf{Game}_{2\text{-}3}$ and the Game_3 are identical until the event fake' occurs, we have $|\Pr[T^b_{2\text{-}3}] - \Pr[T^b_3]| \leq \Pr[\mathsf{fake}']$. Thus, it is enough to show $\Pr[\mathsf{fake}'] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}_4}$. We follow the same way to construct $\mathcal{B}_4$ as seen in the above.

We first modify the game slightly as follows. In the $\mathsf{RepSetup}$ algorithm, we compute $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}^1_0(1^n)$ instead of $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$. Note that its affects nothing about the probability $\mathcal{A}$ wins.

Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[\mathsf{fake}']$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}_4$ against the simulation-extractability of Π_1. The description of $\mathcal{B}_4$ is as follows.

$\mathcal{B}_4$ initially receives crs_1, computes $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n)$, $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$ and $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}^2_0(1^n)$, and sets $\mathsf{tpk} := (\mathsf{ek}_1, \mathsf{ek}_2)$, $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}_4$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk})$. It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to AddU, SndToU, and $\mathsf{AnonChal}_b$, $\mathcal{B}_4$ operates the same way as $\mathcal{B}_3$ does in the above.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}_4$ computes $(C, \tau, \pi_{\mathsf{acc}})$ in the same as defined in Section B.1, queries $(\mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle)$ to the Sim^1_1 oracle, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma_{\mathsf{acc}}) \leftarrow \mathsf{info}$, and receives a proof π. Finally, it returns (C, τ, π) to $\mathcal{A}$.
- For each query to $\mathsf{OTrace}(\mathsf{ipk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$, $\mathcal{B}_4$ makes a response as follows. If the event fake occurs, then $\mathcal{B}_4$ outputs $(\mathsf{M}, (C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau), \widetilde{\pi})$, where $(C, \tau, \widetilde{\pi}) \leftarrow \Sigma$. Otherwise, $\mathcal{B}_4$ returns the same as defined in Section B.1 except that it contains a simulated proof that $\mathcal{B}_4$ can obtain by querying the Sim^1_1 oracle, instead of a real proof.

When $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}_4$ outputs 0 and terminates.

The above completes the description of $\mathcal{B}_4$. Let $\mathsf{W} \leftarrow \mathsf{SimExt}_1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$, where $\mathsf{X} \leftarrow (C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau)$. We note that the event fake' occurs, $(\mathsf{X}, \mathsf{W}) \notin \rho_1 \wedge \mathsf{NIZK.Verify}(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \widetilde{\pi}) = 1$ since we clearly have $\mathsf{X} \notin L_{\rho_1}$, where $L_{\rho_1} := \{\mathsf{X} \mid \exists \mathsf{W}; (\mathsf{X}, \mathsf{W}) \in \rho_1\}$. We need to show $(\mathsf{M}, \mathsf{X}, \widetilde{\pi}) \notin L_{\mathcal{S}}$, where $L_{\mathcal{S}}$ is the list of queries and responses to Sim_1. $\mathcal{A}$ possibly gets a simulated proof $\widetilde{\pi}$ via querying to the OSign or $\mathsf{AnonChal}_b$ oracle. However, in case $\mathcal{B}_4$ queries a statement X to the OSign oracle and it returns $\widetilde{\pi}$, we always have $\mathsf{X} \in L_{\rho_1}$, thus the event fake never occurs. In addition, $\widetilde{\pi}$ is never included in a response from a query to the $\mathsf{AnonChal}_b$ oracle since all responds from the $\mathsf{AnonChal}_b$ oracle are included in CL and the OTrace oracle returns $\perp$ if there exists $(\mathsf{ipk}, -, -, \mathsf{item}, \mathsf{M}, \Sigma) \in \mathsf{CL}$. Therefore, we have $\Pr[\mathsf{fake}'] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}_4}(n)$.

Proof of Lemma D.4.

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $|\Pr[T_3^0]-\Pr[T_3^1]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ attacking the indistinguishability of LIT. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ initially receives $(\mathsf{tagpk}_0, \mathsf{tagpk}_1)$, randomly chooses $r_{\mathsf{PKE}} \leftarrow \{0,1\}^{\mathsf{poly}(n)}$, computes $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{Sim}_0^1(1^n)$, $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}_0^2(1^n)$, $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}$ $(1^n; r_{\mathsf{PKE}})$, and $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$, and sets $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$, $(\mathsf{tpk}, \mathsf{tsk}) := ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$, and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{CL}, \mathsf{SL} := \emptyset$. Moreover, $\mathcal{B}$ chooses two indexes $i^*, j^* \in [Q]$ uniform randomly, where Q is the number of queries to the AddU oracle. Note that we can fix Q before running $\mathcal{A}$ without loss of generality. Then, $\mathcal{B}$ sets $\mathsf{upk}_{i^*} \leftarrow \mathsf{tagpk}_0$ and $\mathsf{upk}_{j^*} \leftarrow \mathsf{tagpk}_1$. We remark that $\mathcal{B}$ cannot compute ssk_{i^*} and ssk_{j^*} since $\mathcal{B}$ does not know the tag secret keys $(\mathsf{tagsk}_0, \mathsf{tagsk}_1)$ corresponding to the tag public keys $(\mathsf{tagpk}_0, \mathsf{tagpk}_1)$, respectively. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk})$. It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to SndToU, OTrace, and $\mathsf{RevUser}$, $\mathcal{B}$ operates the same as defined in Section B.1 with the modification for Game_2^b.
- For each query to $\mathsf{AddU}()$, $\mathcal{B}$ returns tagpk_0 for the i^*-th query and tagpk_1 for the j^*-th query. Other operations are the same as defined in Section B.1.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, if $\mathsf{upk} = \mathsf{tagpk}_0$ (resp., $\mathsf{upk} = \mathsf{tagpk}_1$), then $\mathcal{B}$ queries $(0, \langle \mathsf{ipk}, \mathsf{item}\rangle)$ (resp., $(1, \langle \mathsf{ipk}, \mathsf{item}\rangle)$) to the $\mathcal{O}_{tag}^{\mathsf{ind}}$ oracle, and receives a tag τ. Otherwise, $\mathcal{B}$ computes $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}, \langle \mathsf{ipk}, \mathsf{item}\rangle)$, where usk is the secret key of the user upk. Other operations are the same as defined in Section B.1 except that it contains a simulated proof instead of a real proof.
- For each query to $\mathsf{AnonChal}_b(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M})$, $\mathcal{B}$ makes a response as follows:

  Check: Return $\perp$ and continue if $\mathsf{ssk}_i = \perp$ where $\mathsf{ssk}_i \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}_i][\mathsf{item}]$ for $i = 0$ or 1. Else, output 0 and terminate if $\mathsf{upk}_0 \neq \mathsf{tagpk}_0$ or $\mathsf{upk}_1 \neq \mathsf{tagpk}_1$. Otherwise, continue.

  Compute $c^* = (c_1^*, c_2^*)$: Compute $c_1^* \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, 0^{|\mathsf{upk}_0|})$ and $c_2^* \leftarrow \mathsf{PKE.Enc}$ $(\mathsf{ek}_2, 0^{|\mathsf{upk}_0|})$. Note that we assume $|\mathsf{upk}_0| = |\mathsf{upk}_1|$ w.l.o.g.

  Query: Send $\langle \mathsf{ipk}, \mathsf{item}\rangle$ to the challenger for the indistinguishability of LIT, and receives τ.

  Compute $\widetilde{\pi}$: Compute $\widetilde{\pi} \leftarrow \mathsf{Sim}_1^1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{M}, \mathsf{X})$, where $\mathsf{X} \leftarrow \langle c^*, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle$, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$.

  Finally, it returns $(c^*, \tau, \widetilde{\pi})$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}$ outputs 0 and terminates if $|\mathsf{CL}| \neq 1$ or $\exists(\mathsf{ipk}, \mathsf{upk}_i, \mathsf{item}, -, -) \in \mathsf{SL}$ for $i = 0$ or 1, where $(\mathsf{ipk}, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{item}, \mathsf{M}, \Sigma_b) \leftarrow \mathsf{CL}$. Otherwise, $\mathcal{B}$ outputs b' and terminates.

The above completes the description of $\mathcal{B}$. Let d be the challenge bit for the indistinguishability game of LIT. If $\mathsf{upk}_0 = \mathsf{tagpk}_0$ and $\mathsf{upk}_1 = \mathsf{tagpk}_1$ and $d = 0$, τ is represented as $\tau = \mathsf{LIT.Tag}(\mathsf{tagsk}_0, \langle \mathsf{ipk}, \mathsf{item}\rangle)$. Therefore, $\mathcal{B}$ perfectly simulates Game_4^0 for $\mathcal{A}$. On the other hand, if $\mathsf{upk}_0 = \mathsf{tagpk}_0$ and $\mathsf{upk}_1 = \mathsf{tagpk}_1$ and $d = 1$, τ is represented as $\tau = \mathsf{LIT.Tag}(\mathsf{tagsk}_1, \langle \mathsf{ipk}, \mathsf{item}\rangle)$. Therefore, $\mathcal{B}$ perfectly simulates Game_4^1 for $\mathcal{A}$. Note that in both games, $\mathcal{B}$ never queries $\langle \mathsf{ipk}, \mathsf{item} \rangle$ from the definition of the OSign oracle. Now, the probability that $\mathsf{upk}_0 = \mathsf{tagpk}_0$ and $\mathsf{upk}_1 = \mathsf{tagpk}_1$ hold is $Q(Q-1)$. Thus, we have $1/2 \cdot \left| Q(Q-1) \Pr[T_4^0] - Q(Q-1) \Pr[T_4^1] \right| = \mathsf{Adv}^{\mathsf{tag\text{-}ind}}_{\mathsf{LIT},\mathcal{B}}(n)$. Clearly, $|\Pr[T_4^0] - \Pr[T_4^1]| = \frac{2}{Q(Q-1)} \mathsf{Adv}^{\mathsf{tag\text{-}ind}}_{\mathsf{LIT},\mathcal{B}}(n)$.

D.2 Proof of Theorem 5.3

Proof. Let us fix a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ attacking non-frameability of the ARS and the value of the security parameter n. The attack game used to define non-frameability is defined in Figure 3. Let Game_0 be the original attack game, and let T_0 be the event that $\mathcal{A}$ wins in the Game_0. Our overall strategy for the proof is as follows. We shall define a modified attack game Game_1. Then, we separate the winning condition in Game_1 into two cases: $\mathcal{A}_2$ wins when it outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ such that $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 1 \wedge (-, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma) \notin \mathsf{SL}$ and (1) $\exists(-, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}', \Sigma') \in \mathsf{SL}$ s.t. $\mathsf{Link}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, (\mathsf{M}, \Sigma), (\mathsf{M}', \Sigma')) = 1$, or (2) $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 1$. We denote T_1^1 as the event that $\mathcal{A}$ wins with the former condition, and T_1^2 as the event that $\mathcal{A}$ wins with the latter condition. Each of the games operates on the same underlying probability space. Our strategy is to show that the quantity $|\Pr[T_0] - \Pr[T_1]|$ is negligible and $\Pr[T_1^1]$ and $\Pr[T_1^2]$ are negligible, respectively.

Game_1. We now modify Game_0 to obtain a new game Game_1. These two games are identical, except for a small modification to the $\mathsf{RepSetup}$ algorithm and the OSign oracle. Instead of using the original Sign algorithm to sign a message, we use modified algorithms, in which some steps are replaced by as follows:

- In the $\mathsf{RepSetup}$ algorithm, $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$ is replaced by $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{Sim}_0^1(1^n)$.
- In the Sign algorithm, $\pi \leftarrow \mathsf{NIZK.Prove}_1(\mathsf{crs}_1, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$ is replaced by $\widetilde{\pi} \leftarrow \mathsf{Sim}_1^1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{lbl}, \mathsf{X})$, where $\mathsf{lbl} \leftarrow \mathsf{M}$, $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$, and $\mathsf{W} \leftarrow \langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}, \mathsf{wit}, \theta, r \rangle$.

Any difference between these two games yields a PPT algorithm that distinguishes the real proof from the simulated proof. More precisely, we have:

Lemma D.5. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$|\Pr[T_1] - \Pr[T_0]| \leq \mathsf{Adv}^{\mathsf{zk}}_{\Pi_1,\mathcal{B}}(n). \tag{10}$$

Proof. Assume that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which makes the probability $|\Pr[T_1] - \Pr[T_0]|$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the zero-knowledge property of Π_1 with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ initially receives crs_1, computes $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, and sets $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp})$. After $\mathcal{B}$ receives $(\mathsf{st}, \mathsf{tpk})$ from $\mathcal{A}_1$, it runs $\mathcal{A}_2(\mathsf{st})$ It responds to each of queries from $\mathcal{A}$ as follows:

- For each of queries to AddU and SndToU, $\mathcal{B}$ operates the same as defined in Section B.1.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}$ makes a response as follows:
  Check: Return $\perp$ if $\mathsf{ssk} = \perp$ where $\mathsf{ssk} \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}]$. Otherwise, continue.
  Compute $C = (c_1, c_2)$: Choose r_1 and r_2 uniform randomly, and compute $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2)$.
  Compute τ: Compute $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}, \langle \mathsf{ipk}, \mathsf{item} \rangle)$, where $(\mathsf{upk}, \mathsf{usk}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}}, \theta) \leftarrow \mathsf{ssk}$.
  Compute π_{acc}: Compute $\pi_{\mathsf{acc}} \leftarrow \mathsf{ACC.Prove}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{aux}_{\mathsf{acc}}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}})$, where $\mathsf{pp}_{\mathsf{acc}} \leftarrow \mathsf{ipk}$ and $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$.
  Query: Set $\mathsf{lbl} \leftarrow \mathsf{M}$, $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$, and $\mathsf{W} \leftarrow \langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle$. Then, send $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ to the challenger for the zero-knowledge property of Π_1, and receive π.
  Finally, it returns (C, τ, π) to $\mathcal{A}$ and updates $\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(t^{\mathsf{ipk}}_{\mathsf{cur}}, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma)\}$, where $t^{\mathsf{ipk}}_{\mathsf{cur}}$ is the current epoch.

When $\mathcal{A}_2$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}$ outputs 1 and terminates if $\mathcal{A}$ wins. Otherwise, $\mathcal{B}$ outputs 0 and terminates.

The above completes the description of $\mathcal{B}$. If crs_1 is generated by the $\mathsf{NIZK.Setup}_1$ (resp., Sim^1_0) algorithm and $\mathcal{B}$ accesses the $\mathsf{NIZK.Prove}_1$ (resp., Sim^1_1) oracle, then $\mathcal{B}$ perfectly simulates Game_1 (resp., Game_2) for $\mathcal{A}$. Therefore, we have $|\Pr[T_1] - \Pr[T_0]| = \mathsf{Adv}^{\mathsf{zk}}_{\Pi_1,\mathcal{B}}(n)$.

Next, we show $\Pr[T^1_1] = \mathsf{negl}(n)$. Beforehand, we slightly modify the game as follows: we use $\mathsf{SimExt} = (\mathsf{SimExt}_0, \mathsf{SimExt}_1)$ instead of Sim_1. We note that this modification does not affect the probability $\mathcal{A}$ wins since SimExt_0 outputs the identical distribution to Sim^1_0 with respect to the first two outputs. At the end of Game_1, $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$. Let $\Sigma = (C, \tau, \widetilde{\pi})$ and $\mathsf{X} = \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$. Since $\mathsf{NIZK.Verify}(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \widetilde{\pi}) = 1 \wedge (-, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma) \notin \mathsf{SL}$ when $\mathcal{A}$ wins, we can extract the witness W such that $(\mathsf{X}, \mathsf{W}) \in \rho_1$ by using the extractor SimExt. We prove that for any $\mathcal{A}$ in Game_1, there exists such an extractor. Let E_{fail} be the event that $(\mathsf{X}, \mathsf{W}) \notin \rho_1$, i.e., SimExt fails to extract. Clearly, we have $\Pr[T^1_1] \leq \Pr[T^1_1 \wedge E_{\mathsf{fail}}] + \Pr[T^1_1 \wedge \neg E_{\mathsf{fail}}]$. We will prove $\Pr[T^1_1 \wedge E_{\mathsf{fail}}] = \mathsf{negl}(n)$ and $\Pr[T^1_1 \wedge \neg E_{\mathsf{fail}}] = \mathsf{negl}(n)$ in order.

Lemma D.6. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[T^1_1 \wedge E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}}(n). \tag{11}$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which causes the event T^1_1 and E_{fail}. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the simulation extractability of Π_1, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ initially receives crs_1, computes $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, and sets $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}_1(\mathsf{pp})$. After $\mathcal{B}$ receives $(\mathsf{st}, \mathsf{tpk})$ from $\mathcal{A}_1$, it runs $\mathcal{A}_2(\mathsf{st})$. It responds to each of queries from $\mathcal{A}$ by the same way as the proof of Lemma D.5 except that it sends a query to the Sim_1 oracle instead of the challenger for the ZK property of Π_1 when responding to each OSign query.

When $\mathcal{A}_2$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}$ outputs $(\mathsf{M}, \mathsf{X}, \widetilde{\pi})$ and terminates if $\mathcal{A}$ wins, where $\Sigma = (C, \tau, \widetilde{\pi})$ and $\mathsf{X} = \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle$. Otherwise, $\mathcal{B}$ outputs 0 and terminates.

The above completes the description of $\mathcal{B}$. Let $\mathsf{W} \leftarrow \mathsf{SimExt}_1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$. We note that when $\mathcal{A}$ wins and the event E_{fail} occurs, $(\mathsf{M}, \mathsf{X}, \widetilde{\pi}) \notin L_{\mathcal{S}} \wedge (\mathsf{X}, \mathsf{W}) \notin \rho_1 \wedge \mathsf{NIZK.Verify}(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \widetilde{\pi}) = 1$, where $L_{\mathcal{S}}$ is the list of queries and responses to Sim_1. Therefore, we have $\Pr[T_1^1 \wedge E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}}(n)$.

In the following, we assume that the SimExt successfully extract the witness. Let $\mathsf{W} = \langle \mathsf{upk}^*, \mathsf{usk}^*, \mathsf{id}, \pi_{\mathsf{ACC}}, \theta, r_1, r_2\rangle$ be the witness extracted from the proof $\widetilde{\pi}$. We further separate $\mathcal{A}$'s winning condition into two cases: (i) the extracted tag public key $\mathsf{tagpk}^* := \mathsf{upk}^*$ is the same as upk, i.e., $\mathsf{tagpk}^* = \mathsf{upk}$; (ii) otherwise. We denote $T_1^{1,1}$ as the event that $\mathcal{A}$ wins in the case (i) and $T_1^{1,2}$ as the event $\mathcal{A}$ wins in the case (ii). Clearly, we have $\Pr[T_1^1 \wedge \neg E_{\mathsf{fail}}] \leq \Pr[T_1^{1,1} \wedge \neg E_{\mathsf{fail}}] + \Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}}]$. In the following, we show $\Pr[T_1^{1,1} \wedge \neg E_{\mathsf{fail}}] \leq \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{B}_1}(n)$ and $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{link\text{-}sound}}_{\mathsf{LIT},\mathcal{B}_2}(n)$, respectively.

Lemma D.7. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[T_1^{1,1} \wedge \neg E_{\mathsf{fail}}] = \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{B}}(n). \tag{12}$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_1^{1,1} \wedge \neg E_{\mathsf{fail}}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the key-secrecy of LIT with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, initially given tagpk, computes $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}_0(1^n)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, and sets $\mathsf{pp} := (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}_1(\mathsf{pp})$. After $\mathcal{B}$ receives $(\mathsf{st}, \mathsf{tpk})$ from $\mathcal{A}_1$, it runs $\mathcal{A}_2(\mathsf{st})$. Let the number of queries to AddU from $\mathcal{A}$ be $K = \mathsf{poly}(n)$. $\mathcal{B}$ randomly chooses $k \leftarrow [K]$ and sets the k-th user's key pair $(\mathsf{upk}_k, \mathsf{usk}_k)$ (corresponding to the k-th query to AddU) as the target, i.e., $\mathcal{B}$ sets $\mathsf{upk}_k = \mathsf{tagpk}$ and does not know usk_k. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ as follows.

- For each query to AddU, $\mathcal{B}$ operates the same as defined in Section B.1 except that it returns upk_k for the k-th query from $\mathcal{A}$.
- For each query to SndToU, $\mathcal{B}$ operates the same as defined in Section B.1.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}$ makes a response as follows:
  Check: Return $\perp$ if $\mathsf{ssk} = \perp$ where $\mathsf{ssk} \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}]$. Otherwise, continue.
  Compute $C = (c_1, c_2)$: Choose r_1 and r_2 uniform randomly, and compute $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2)$.
  Compute τ: If $\mathsf{upk} = \mathsf{upk}_k$, then send $\langle \mathsf{ipk}, \mathsf{item}\rangle$ to the tag generation oracle $\mathcal{O}^{\mathsf{sec}}_{tag}$, and receive τ. Otherwise, compute $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}, \langle \mathsf{ipk}, \mathsf{item}\rangle)$, where $(\mathsf{upk}, \mathsf{usk}, \mathsf{wit}) \leftarrow \mathsf{ssk}$.

Compute $\widetilde{\pi}$: Compute $\widetilde{\pi} \leftarrow \mathsf{Sim}_1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{M}, \mathsf{X})$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$.

Finally, $\mathcal{B}$ returns $(C, \tau, \widetilde{\pi})$ to $\mathcal{A}$ and updates $\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(t^{\mathsf{ipk}}_{\mathsf{cur}}, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma)\}$, where $t^{\mathsf{ipk}}_{\mathsf{cur}}$ is the current epoch.

When $\mathcal{A}_2$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}$ outputs $\perp$ and terminates if $\mathsf{upk} \neq \mathsf{upk}_k$. Otherwise, $\mathcal{B}$ parses $(C, \tau, \widetilde{\pi}) \leftarrow \Sigma$ and computes $\langle \mathsf{upk}^*, \mathsf{usk}^*, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle \leftarrow \mathsf{SimExt}_1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$. Then, $\mathcal{B}$ outputs usk^* and terminates.

The above completes the description of $\mathcal{B}$. By the definition of the event $T_1^{1,1} \wedge \neg E_{\mathsf{fail}}$, we have $\mathsf{upk} = \mathsf{upk}^*$ and $\mathsf{LIT.ChkKey}(\mathsf{upk}^*, \mathsf{usk}^*) = 1$. The probability that $\mathsf{upk} = \mathsf{upk}_k$ is $1/K$. Therefore, we have $\Pr[T_1^{1,1} \wedge \neg E_{\mathsf{fail}}] = K \cdot \mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{B}}(n)$.

Lemma D.8. *There exists a PPT algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ such that*

$$\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{key\text{-}robust}}_{\mathsf{LIT},\mathcal{B}_1}(n) + \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{B}_2}(n). \tag{13}$$

Proof. Let $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ be the output of $\mathcal{A}$ in Game_1 and usk be the secret key corresponding to upk. Since the event E_{fail} never occurs, we can extract a witness $\mathsf{W} = \langle \mathsf{upk}^*, \mathsf{usk}^*, \pi_{\mathsf{ACC}}, r \rangle$ from the proof π included in Σ. We divide the case into two depending on whether $\mathsf{usk} \neq \mathsf{usk}^*$ or not. Intuitively, in case $\mathsf{usk} = \mathsf{usk}^*$, $\mathcal{A}$ breaks the key-secrecy of LIT since we can extract the corresponding secret key usk by extracting the witness. In other case, $\mathcal{A}$ breaks the key-robustness of LIT. We denote the latter case by the event E_{bad}. We clearly have $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}}] \leq \Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}] + \Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}]$. We show each probability is negligible in order.

First, we show $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}] \leq \mathsf{Adv}^{\mathsf{key\text{-}robust}}_{\mathsf{LIT},\mathcal{B}_1}(n)$. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}_1$ that breaks the key-robustness of LIT with non-negligible probability.

$\mathcal{B}_1$ computes $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}_0(1^n)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}_1$ runs $\mathcal{A}_1(\mathsf{pp})$. After $\mathcal{B}_1$ receives $(\mathsf{st}, \mathsf{tpk})$ from $\mathcal{A}_1$, it runs $\mathcal{A}_2(\mathsf{st})$. $\mathcal{B}_1$ responds to each of queries from $\mathcal{A}_2$ as follows.

- For each of queries to AddU and SndToU, $\mathcal{B}_1$ operates the same as defined in Section B.1.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}_1$ operates the same as defined in Section B.1 except that it returns the simulated proof $\widetilde{\pi} \leftarrow \mathsf{Sim}_1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{M}, \mathsf{X})$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \tau \rangle$, instead of the real proof.

When $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}_1$ parses $(C, \tau^*, \widetilde{\pi}) \leftarrow \Sigma$ and computes $\langle \mathsf{upk}^*, \mathsf{usk}^*, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle \leftarrow \mathsf{SimExt}_1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau^* \rangle$. Now, there exists $(-, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}', \Sigma') \in \mathsf{SL}$. Let $(C', \tau', \pi') \leftarrow \Sigma'$ and usk be the tag secret key corresponding to upk. Note that $\mathcal{B}_1$ knows usk since it generates all users for $\mathcal{A}$. $\mathcal{B}_1$ outputs $(\langle \mathsf{ipk}, \mathsf{item} \rangle, \mathsf{upk}, \mathsf{usk}, \tau', \mathsf{upk}^*, \mathsf{usk}^*, \tau^*)$ and terminates.

The above completes the description of $\mathcal{B}_1$. Since the tag τ' is honestly generated, we have $\mathsf{LIT.ChkTag}(\mathsf{upk}, \mathsf{usk}, \langle \mathsf{ipk}, \mathsf{item} \rangle, \tau') = 1$. Since the event E_{fail} never occurs (i.e., succeed in extracting the witness), we have $\mathsf{LIT.ChkTag}(\mathsf{upk}^*, \mathsf{usk}^*, \langle \mathsf{ipk}, \mathsf{item} \rangle, \tau^*) = 1$. By the definition of the event E_{bad}, we have $\mathsf{usk} \neq \mathsf{usk}^*$. By the definition of the event $T_1^{1,2}$, we have $\mathsf{Link}(\tau', \tau^*) = 1$. Therefore, $\mathcal{B}_1$ breaks the key-robustness of LIT, so we have $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}] \leq \mathsf{Adv}^{\mathsf{key\text{-}robust}}_{\mathsf{LIT},\mathcal{B}_1}(n)$.

Second, we show $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}] \leq \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{B}_2}(n)$. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}_2$ that breaks the key-secrecy of LIT with non-negligible probability. The description of $\mathcal{B}_2$ is as follows.

$\mathcal{B}_2$, initially given tagpk, computes $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}_0(1^n)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} = \emptyset$. Then, it runs $\mathcal{A}_1(\mathsf{pp})$. After $\mathcal{B}_2$ receives $(\mathsf{st}, \mathsf{tpk})$, it runs $\mathcal{A}_2(\mathsf{st})$. Let the number of queries to AddU from $\mathcal{A}$ be $K = \mathsf{poly}(n)$. $\mathcal{B}$ randomly chooses $k \leftarrow [K]$ and sets the k-th user's key pair $(\mathsf{upk}_k, \mathsf{usk}_k)$ (corresponding to the k-th query to AddU) as the target, i.e., $\mathcal{B}$ sets $\mathsf{upk}_k = \mathsf{tagpk}$ and does not know usk_k.

$\mathcal{B}_2$ responds to each of queries from $\mathcal{A}$ as follows.

- For each query to AddU, $\mathcal{B}_2$ operates the same as defined in Section B.1 except that it returns tagpk for the k-th query from $\mathcal{A}$.
- For each query to SndToU, $\mathcal{B}_2$ operates the same as defined in Section B.1.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}$ makes a response as follows:
  Check: Return $\perp$ if $\mathsf{ssk} = \perp$ where $\mathsf{ssk} \leftarrow \mathsf{HSSKL}[\mathsf{ipk}][\mathsf{upk}][\mathsf{item}]$. Otherwise, continue.
  Compute $C = (c_1, c_2)$: Choose r_1 and r_2 uniform randomly, and compute $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2)$.
  Compute τ: If $\mathsf{upk} = \mathsf{upk}_k$, then send $\langle \mathsf{ipk}, \mathsf{item} \rangle$ to the tag generation oracle $\mathcal{O}^{\mathsf{sec}}_{tag}$, and receive τ. Otherwise, compute $\tau \leftarrow \mathsf{LIT.Tag}(\mathsf{usk}, \langle \mathsf{ipk}, \mathsf{item} \rangle)$, where $(\mathsf{upk}, \mathsf{usk}, \mathsf{id}, \mathsf{wit}_{\mathsf{id}}) \leftarrow \mathsf{ssk}$.
  Compute $\widetilde{\pi}$: Compute $\widetilde{\pi} \leftarrow \mathsf{Sim}_1(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{M}, \mathsf{X})$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$.
  Finally, $\mathcal{B}$ returns $(C, \tau, \widetilde{\pi})$ to $\mathcal{A}$ and updates $\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(t^{\mathsf{ipk}}_{\mathsf{cur}}, \mathsf{ipk}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma)\}$, where $t^{\mathsf{ipk}}_{\mathsf{cur}}$ is the current epoch.

When $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}_2$ outputs $\perp$ and terminates if $\mathsf{upk} \neq \mathsf{tagpk}$. Otherwise, $\mathcal{B}_2$ parses $(C, \tau^*, \widetilde{\pi}) \leftarrow \Sigma$ and computes $\langle \mathsf{upk}^*, \mathsf{usk}^*, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle \leftarrow \mathsf{SimExt}_1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau^* \rangle$. Finally, $\mathcal{B}_2$ outputs usk^* and terminates.

The above completes the description of $\mathcal{B}_2$. Since the event E_{bad} never occurs, the extracted secret key usk^* is corresponding to upk, i.e., $\mathsf{LIT.ChkKey}(\mathsf{upk}, \mathsf{usk}^*) = 1$. The probability that $\mathsf{upk} = \mathsf{tagpk}$ is $1/K$. Therefore, we have $\Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}] = K \cdot \mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{B}_2}(n)$.

Clearly, we have the following since we can separate the condition $\mathcal{A}$ wins into two case.

$$\Pr[T_1] \leq \Pr[T_1^1] + \Pr[T_1^2]. \tag{14}$$

$$\Pr[T_1^1] \leq \Pr[T_1^1 \wedge E_{\mathsf{fail}}] + \Pr[T_1^{1,1} \wedge \neg E_{\mathsf{fail}}] + \Pr[T_1^{1,2} \wedge \neg E_{\mathsf{fail}}]. \tag{15}$$

Finally, we show $\Pr[T_1^2] = \mathsf{negl}(n)$. Consider the case that $\mathcal{A}_2$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$. Let $\Sigma = (C, \tau, \widetilde{\pi})$, $\mathsf{W} \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$, and $(\mathsf{upk}^*, \mathsf{usk}^*)$ is included in W. In addition, let R be the event that $\mathsf{upk} \neq \mathsf{upk}^*$ and $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 1$. Clearly, we have $\Pr[T_1^2] \leq \Pr[R] + \Pr[T_1^2 \wedge \neg R]$. We will show both $\Pr[R]$ and $\Pr[T_1^2 \wedge \neg R]$ are negligible in n.

Lemma D.9. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[R] = \mathsf{Adv}_{\Pi_2, \mathcal{B}}^{\mathsf{sound}}(n). \tag{16}$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which causes the event R. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the soundness of Π_2, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ initially receives crs_2, computes $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}_0(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}_1(\mathsf{pp})$. After it receives $(\mathsf{st}, \mathsf{tpk})$ from $\mathcal{A}_1$, it runs $\mathcal{A}(\mathsf{st})$. It responds to each of queries from $\mathcal{A}_2$ as follows.

- For each of queries to AddU and SndToU, $\mathcal{B}$ operates the same as defined in Section B.1.
- For each query to $\mathsf{OSign}(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M})$, $\mathcal{B}$ operates the same as defined in Section B.1 except that it returns the simulated proof instead of the real proof.

When $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}$ outputs $(\mathsf{M}, \langle \mathsf{ek}_1, c_1 \mathsf{upk} \rangle, \Pi_{\mathsf{Trace}})$ and terminates, where $((c_1, c_2), \tau, \widetilde{\pi}) \leftarrow \Sigma$.

The above completes the description of $\mathcal{B}$. By the definition of the event R, we have $\mathsf{upk}^* \neq \mathsf{upk} \wedge \mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 1$, where upk^* is included in $\mathsf{W} \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$ and $\mathsf{X} = \langle (c_1, c_2), \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$. Now, $c_1 = \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}^*; r_1)$ holds since $(\mathsf{X}, \mathsf{W}) \in \rho_1$, so we have $\langle \mathsf{ek}_1, c_1, \mathsf{upk} \rangle \notin L_{\rho_2}$. On the other hand, $\mathsf{NIZK.Verify}_2(\mathsf{crs}_2, \mathsf{M}, \langle \mathsf{ek}_1, c_1, \mathsf{upk} \rangle, \Pi_{\mathsf{Trace}}) = 1$ since $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 1$. Therefore, $\mathcal{B}$ has a successful attack. Thus, we have $\Pr[R] \leq \mathsf{Adv}_{\Pi_2, \mathcal{B}}^{\mathsf{sound}}(n)$.

Lemma D.10. *There exist PPT algorithms $\mathcal{B}$ such that*

$$\Pr[T_1^2 \wedge \neg R] = \mathsf{poly}(n) \cdot \mathsf{Adv}_{\mathsf{LIT}, \mathcal{B}}^{\mathsf{key\text{-}sec}}(n). \tag{17}$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which makes the probability $\Pr[T_1^2]$ non-negligible. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the key-secrecy of LIT with non-negligible probability, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ initially receives tagpk, computes $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}_0(1^n)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HUL}, \mathsf{HSSKL}, \mathsf{SL} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}_1(\mathsf{pp})$. After it receives $(\mathsf{st}, \mathsf{tpk})$ from $\mathcal{A}_1$, it runs $\mathcal{A}_2(\mathsf{st})$. Let the number of queries to AddU from $\mathcal{A}_2$ be $K = \mathsf{poly}(n)$. $\mathcal{B}$ randomly chooses $k \leftarrow [K]$ and sets the k-th user's key pair $(\mathsf{upk}_k, \mathsf{usk}_k)$ (corresponding to the k-th query to AddU) as the target tagpk, i.e., $\mathcal{B}$ is given $\mathsf{upk}_k = \mathsf{tagpk}$ and tries to break the key-secrecy.

$\mathcal{B}$ responds to each of queries from $\mathcal{A}_2$ in the same way as $\mathcal{B}$ does in the proof of Lemma D.7, so we omit it here. When $\mathcal{A}_2$ outputs $(\mathsf{ipk}, \mathsf{info}, \mathsf{upk}, \mathsf{item}, \mathsf{M}, \Sigma, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}$ outputs $\perp$ and terminates if $\mathsf{upk} \neq \mathsf{upk}_k$. Otherwise, $\mathcal{B}$ parses $(C, \tau, \widetilde{\pi}) \leftarrow \Sigma$ and computes $\langle \mathsf{upk}^*, \mathsf{usk}^*, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle \leftarrow \mathsf{SimExt}(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \widetilde{\pi})$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$. Then, $\mathcal{B}$ outputs usk^* and terminates.

The above completes the description of $\mathcal{B}$. By the definition of the event $T_1^2 \wedge \neg R$, we have $\mathsf{upk}^* = \mathsf{upk}$ since $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 1$. Thus, we have $\mathsf{LIT.ChkKey}(\mathsf{upk}, \mathsf{usk}^*) = 1$, which means that $\mathcal{B}$ has a successful attack. Now, the probability that $\mathsf{upk} = \mathsf{upk}_k$ is $1/K$. Therefore, we have $\Pr[T_1^{1,2}] = K \cdot \mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{B}}(n)$.

Theorem 5.3 now follows immediately from (10)-(17).

D.3 Proof of Theorem 5.4

Proof. Let $\mathcal{A}$ be a PPT adversary in the experiment $\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{ARS},\mathcal{A}}(n)$ defined in Figure 4. We can separate the case $\mathcal{A}$ wins into two cases: when $\mathcal{A}$ outputs $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$, $\mathcal{A}$ wins if $|\mathsf{HKIL}| = 1$, $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 1$, and (i) $t \notin \mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}]$, or (ii) $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 0$, where t is included in info and $(\mathsf{upk}, \Pi_{\mathsf{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$. We denote E as the event that $\mathcal{A}$ wins with the condition (i) and F as the event that $\mathcal{A}$ wins with the condition (ii). Then, we have $\mathsf{Adv}^{\mathsf{trace}}_{\mathsf{ARS},\mathcal{A}}(n) \leq \Pr[E] + \Pr[F]$. We will analyze the probability that each event occurs in order.

First, we analyze the probability $\Pr[E]$. Let $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$ be the output by $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{ARS},\mathcal{A}}(n)$ and info includes an epoch t. When the event E occurs, we have $t \notin \mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}]$. In addition, let E_{fail} be the event that the knowledge extractor Ext fails to extract the witness from the proof π included in Σ. We clearly have $\Pr[E] \leq \Pr[E \wedge E_{\mathsf{fail}}] + \Pr[E \wedge \neg E_{\mathsf{fail}}]$. We will prove that each probability is negligible in n in order.

Beforehand, we slightly modify the game as follows: we use $\mathsf{SimExt}_1 = (\mathsf{SimExt}_0^1, \mathsf{SimExt}_1^1)$ instead of $\mathsf{NIZK.Setup}_1$. We note that this has negligible effect on the probability $\mathcal{A}$ wins as seen in the proof of appendix D.2.

Lemma D.11. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[E \wedge E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}}(n). \tag{18}$$

Proof. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E \wedge E_{\mathsf{fail}}$. Then, we can construct another PPT adversary $\mathcal{B}$ attacking the knowledge soundness of Π_1. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, given crs_1, randomly chooses $r_{\mathsf{PKE}} \leftarrow \{0,1\}^{\mathsf{poly}(n)}$, computes $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})$, and $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$, $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$, and $\mathsf{HKIL}, \mathsf{IsActive} =$

$\emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk}, \mathsf{tsk})$. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ in the same manner as defined in Section B.1. Note that $\mathcal{B}$ can respond correctly to all queries since it generates all parameters for ARS except crs_1, which is honestly generated by the challenger for the knowledge soundness of $\mathit{\Pi}_1$. When $\mathcal{A}$ outputs $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \mathit{\Sigma})$ and terminates, $\mathcal{B}$ outputs $\bot$ if $|\mathsf{HKIL}| \neq 1$. Otherwise, it outputs $(\mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle, \pi)$ and terminates, where $\mathsf{ipk} \leftarrow \mathsf{HKIL}$ and $(C, \tau, \pi) \leftarrow \mathit{\Sigma}$ and $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$.

The above completes the description of $\mathcal{B}$. When the event E occurs (i.e., $\mathcal{A}$ wins the game), we have $\mathsf{NIZK.Verify}(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \pi) = 1$. On the other hand, when the event E_{fail} occurs (i.e., the extractor Ext fails to extract a witness), we have $(\mathsf{X}, \mathsf{W}) \notin \rho_1$, where $\mathsf{W} \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \mathsf{X}, \pi)$. In addition, $\mathcal{B}$ never queried to the zero-knowledge simulation oracle clearly, i.e., $L_{\mathcal{S}} = \emptyset$. Therefore, when the both events occur, $\mathcal{B}$ has a successful attack on the simulation extractability of $\mathit{\Pi}_1$, i.e., $\Pr[E \wedge E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{se}}_{\mathit{\Pi}_1, \mathcal{B}}(n)$.

Lemma D.12. *There exist PPT algorithms $\mathcal{B}_1, \mathcal{B}_2$ such that*

$$\Pr[E \wedge \neg E_{\mathsf{fail}}] \leq \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ACC}, \mathcal{B}_1}(n) + 2 \cdot \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG}, \mathcal{B}_2}(n). \tag{19}$$

Proof. Let $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \mathit{\Sigma})$ be the output of $\mathcal{A}$ in the experiment. Since the event E_{fail} never occurs, we can extract the witness $\langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}^*, \mathsf{wit}^*_{\mathsf{id}}, \theta^*, r_1, r_2\rangle$ from π included in $\mathit{\Sigma}$. We divide the case into two depending on whether the following three conditions are satisfied: (a) the extracted upk is included in the queries from $\mathcal{A}$ to the $\mathsf{SndToKI}$ oracle, (b) the extracted id^* equals to $\mathsf{id}_{\mathsf{upk}}$ that is recorded in $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$, and (c) $\mathcal{A}$ outputs $\mathsf{info} = (t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$ such that it obtained $\langle t, \mathsf{acc}\rangle$ by querying to the $\mathsf{SndToKI}$ or $\mathsf{RevUser}$ oracle. Roughly, in case the above three conditions are satisfied, $\mathcal{A}$ breaks the soundness of ACC. Otherwise, $\mathcal{A}$ breaks the EUF-CMA security of SIG. We denote the former case by the event E_{bad}. We clearly have $\Pr[E \wedge \neg E_{\mathsf{fail}}] \leq \Pr[E \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}] + \Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}]$. We analyze each probability in order.

Firstly, we show $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}] \leq \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ACC}, \mathcal{B}_1}(n)$. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E_1 \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}$. Then, we can construct another PPT adversary $\mathcal{B}_1 = (\mathcal{B}_{1,1}, \mathcal{B}_{1,2})$ attacking the soundness of ACC. The description of $\mathcal{B}_1$ is as follows.

$\mathcal{B}_{1,1}$, given $\mathsf{pp}_{\mathsf{acc}}$, randomly chooses $r_{\mathsf{PKE}} \leftarrow \{0,1\}^{\mathsf{poly}(n)}$, computes $(\mathsf{crs}_1, \mathsf{td}, \xi) \leftarrow \mathsf{SimExt}_0^1(1^n)$, $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})$, and $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$, $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$, and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Then, $\mathcal{B}_{1,1}$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk}, \mathsf{tsk})$. Let the number of queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$ from $\mathcal{A}$ be $K = \mathsf{poly}(n)$. $\mathcal{B}_{1,1}$ randomly chooses $k \leftarrow [K]$ and outputs the k-th set of revoked users R_k to the challenger to the soundness game. $\mathcal{B}_{1,1}$ responds to each of queries from $\mathcal{A}$ as follows.

- For each query to $\mathsf{AddKI}()$, $\mathcal{B}_{1,1}$ computes $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KG}(1^n)$, sets $\mathsf{ipk} \leftarrow (\mathsf{vk}, \mathsf{pp}_{\mathsf{acc}})$ and $\mathbb{I}^{\mathsf{ipk}} := \emptyset$, returns ipk to $\mathcal{A}$, and updates $\mathsf{HKIL} \leftarrow \mathsf{HKIL} \cup \{\mathsf{ipk}\}$ for the first time. If this oracle is called the second time, $\mathcal{B}_{1,1}$ returns $\bot$ and terminates.

- For each query to $\mathsf{SndToKI}(\mathsf{ipk}, \mathsf{upk}, \mathsf{item})$, $\mathcal{B}_{1,1}$ computes in the following steps:
  1. If $t = 0$, then adds $\mathbb{I}^{\mathsf{ipk}} \leftarrow \mathbb{I}^{\mathsf{ipk}} \cup \{\mathsf{item}\}$;
  2. If $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$ is empty, then randomly chooses $\mathsf{id}_{\mathsf{upk}} \leftarrow \{\mathsf{id} \in \{0,1\}^n \mid \text{Does not appear in } \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}\}$ and sets $List_{\mathsf{upk}} \leftarrow \epsilon$. Otherwise, let $(\mathsf{id}_{\mathsf{upk}}, \mathtt{status}, List_{\mathsf{upk}}) \leftarrow \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$;
  3. If $\mathtt{status} = \mathtt{revoked}$ or $\exists(\mathsf{item}, -) \in List_{\mathsf{upk}}$, then returns $\bot$ to $\mathcal{A}$. Otherwise, continues;
  4. Sets a set of revoked users' id R as follows:
     $R := \{\mathsf{id} \in \{0,1\}^n \mid \exists \mathsf{upk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}] = (\mathsf{id}, \mathtt{revoke}, -)\}$;
  5. If it is the k-th of the total number of queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$, then outputs (R, st) and terminates, where $\mathsf{st} \leftarrow (\mathsf{pp}, \mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$. Otherwise, computes $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$, $\theta \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \langle \mathsf{upk}, \mathsf{id}, \mathsf{item}\rangle)$, and $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \langle t+1, \mathsf{acc}\rangle)$, where t is included in $\mathsf{info}_t^{\mathsf{ipk}}$.
  6. Queries $\mathcal{O}_{\mathsf{wit}}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{sk}_{\mathsf{acc}}, \mathsf{id})$ and receives $\mathsf{wit}_{\mathsf{id}}$.

  Then, it returns $(\mathsf{id}_{\mathsf{upk}}, \mathsf{wit}_{\mathsf{id}}, \theta)$ and $\mathsf{info}_{t+1}^{\mathsf{ipk}} := (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$, and updates $\mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}] = [0, \infty]$.
- For each query to $\mathsf{RevUser}(\mathsf{upk})$, $\mathcal{B}_{1,1}$ computes in the following steps:
  1. Gets $(\mathsf{id}_{\mathsf{upk}}, -, -) \leftarrow \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$;
  2. Sets a set of revoked users' id R as follows:
     $R := \{\mathsf{id} \in \{0,1\}^n \mid \exists \mathsf{upk}, \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}] = (\mathsf{id}, \mathtt{revoked}, -)\} \cup \{\mathsf{id}_{\mathsf{upk}}\}$;
  3. If it is the k-th of the total number of queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$, then outputs (R, st) and terminates, where $\mathsf{st} \leftarrow (\mathsf{pp}, \mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$. Otherwise, computes $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$ and $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \langle t+1, \mathsf{acc}\rangle)$, where t is included in $\mathsf{info}_t^{\mathsf{ipk}}$.

  Finally, it updates $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}] \leftarrow (\mathsf{id}_{\mathsf{upk}}, \mathtt{revoked}, -)$ and returns $\mathsf{info}_{t+1}^{\mathsf{ipk}} := (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$.
- For each query to $\mathsf{RtrKIReg}(\mathsf{ipk})$, $\mathcal{B}_{1,1}$ returns $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}$ to $\mathcal{A}$.

The above completes the description of $\mathcal{B}_{1,1}$. Then, $\mathcal{B}_{1,2}$ receives $(\mathsf{st}, \mathsf{acc}_k, \mathsf{aux}_{\mathsf{acc}})$ from the challenger and continues to run $\mathcal{A}$. $\mathcal{B}_{1,2}$ responds to each of queries as the same ways as $\mathcal{B}_{1,1}$ does.

When $\mathcal{A}$ outputs $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$ and terminates, $\mathcal{B}_{1,2}$ outputs $\bot$ and terminates if $\langle t, \mathsf{acc}\rangle \neq \langle t_k, \mathsf{acc}_k\rangle$, where acc is included in info. Otherwise, it computes $(\mathsf{upk}, \mathsf{usk}, \mathsf{id}^*, \pi_{\mathsf{acc}}^*, \theta, r_1, r_2) \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle, \pi)$, where $(C, \tau, \pi) \leftarrow \Sigma$, and outputs $(\mathsf{id}^*, \pi_{\mathsf{acc}}^*)$ and terminates.

The above completes the description of $\mathcal{B}_{1,2}$. Let $(\mathsf{upk}_{\mathsf{Trace}}, \Pi_{\mathsf{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{tsk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$. From the definition of the Trace algorithm, we have $\mathsf{upk}_{\mathsf{Trace}} = \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$. Thus, $\mathsf{upk}_{\mathsf{Trace}} = \mathsf{upk}$ holds since the event E_{fail} never occurs. When the event $E \wedge E_{\mathsf{bad}}$ occurs, we have (i) $\mathsf{id}^* = \mathsf{id}_{\mathsf{upk}}$, (ii) acc included in the output of $\mathcal{A}$ is also included in one of responses from queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$, and (iii) $t \notin \mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}]$, which means $\mathsf{id}_{\mathsf{upk}} \in R$, where $(\mathsf{id}_{\mathsf{upk}}, -, -) \leftarrow \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$. The probability that $\mathsf{acc} = \mathsf{acc}_k$ is $1/K$. In case $\mathsf{acc} = \mathsf{acc}_k$, we have $\mathsf{ACC.Verify}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{acc}_k, \mathsf{id}^*, \pi^*) = 1$ since the Verify algorithm returns 1. Therefore, $\mathcal{B}_1$ has a successful attack on the soundness of ACC with probability $1/K$, i.e., $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge E_{\mathsf{bad}}] \leq K \cdot \mathsf{Adv}_{\mathsf{ACC},\mathcal{B}_1}^{\mathsf{sound}}(n)$.

Secondly, we show $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}] \leq 2 \cdot \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG}\mathcal{B}_2}(n)$. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E_1 \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}$. Then, we can construct another PPT adversary $\mathcal{B}_2$ attacking the EUF-CMA security of SIG. The description of $\mathcal{B}_2$ is as follows.

$\mathcal{B}_2$, given vk, randomly chooses $r_{\mathsf{PKE}} \leftarrow \{0,1\}^{\mathsf{poly}(n)}$, computes $(\mathsf{crs}_1, \mathsf{td}, \xi) \leftarrow \mathsf{SimExt}^1_0(1^n)$, $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})$, and $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$, $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$, and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Then, $\mathcal{B}_2$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk}, \mathsf{tsk})$. $\mathcal{B}_2$ responds to each of queries from $\mathcal{A}$ as follows.

- For each query to $\mathsf{AddKI}()$, $\mathcal{B}_2$ computes $(\mathsf{pp}_{\mathsf{acc}}, \mathsf{sk}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Setup}(1^n)$, sets $\mathsf{ipk} \leftarrow (\mathsf{vk}, \mathsf{pp}_{\mathsf{acc}})$ and $\mathbb{I}^{\mathsf{ipk}} := \emptyset$, returns ipk to $\mathcal{A}$, and updates $\mathsf{HKIL} \leftarrow \mathsf{HKIL} \cup \{\mathsf{ipk}\}$ for the first time. If this oracle is called the second time, $\mathcal{B}_{1,1}$ returns $\perp$ and terminates.
- For each query to $\mathsf{SndToKI}(\mathsf{ipk}, \mathsf{upk}, \mathsf{item})$, $\mathcal{B}_{1,1}$ computes in the following steps:
  1. If $t = 0$, then adds $\mathbb{I}^{\mathsf{ipk}} \leftarrow \mathbb{I}^{\mathsf{ipk}} \cup \{\mathsf{item}\}$;
  2. If $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$ is empty, then randomly chooses $\mathsf{id}_{\mathsf{upk}} \leftarrow \{\mathsf{id} \in \{0,1\}^n \mid \text{Does not appear in } \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}\}$ and sets $List_{\mathsf{upk}} \leftarrow \epsilon$. Otherwise, let $(\mathsf{id}_{\mathsf{upk}}, \mathtt{status}, List_{\mathsf{upk}}) \leftarrow \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$;
  3. If $\mathtt{status} = \mathtt{revoked}$ or $\exists(\mathsf{item}, -) \in List_{\mathsf{upk}}$, then returns $\perp$ to $\mathcal{A}$. Otherwise, continues;
  4. Sets a set of revoked users' id R as follows:
     $R := \{\mathsf{id} \in \{0,1\}^n \mid \exists \mathsf{upk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}] = (\mathsf{id}, \mathtt{revoke}, -)\}$;
  5. Computes $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$ and $\mathsf{wit}_{\mathsf{id}} \leftarrow \mathsf{ACC.Wit}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{sk}_{\mathsf{acc}}, \mathsf{id})$.
  6. Queries $\mathcal{O}_{sign}(\langle \mathsf{upk}, \mathsf{id}_{\mathsf{upk}}, \mathsf{item}\rangle)$ and $\mathcal{O}_{sign}(\langle t+1, \mathsf{acc}\rangle)$ and receives θ and σ, respectively, where t is included in $\mathsf{info}^{\mathsf{ipk}}_t$.

  Then, it returns $(\mathsf{id}_{\mathsf{upk}}, \mathsf{wit}_{\mathsf{id}}, \theta)$ and $\mathsf{info}^{\mathsf{ipk}}_{t+1} := (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$, and updates $\mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}] = [0, \infty]$.
- For each query to $\mathsf{RevUser}(\mathsf{upk})$, $\mathcal{B}_2$ computes in the following steps:
  1. Gets $(\mathsf{id}_{\mathsf{upk}}, -, -) \leftarrow \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$;
  2. Sets a set of revoked users' id R as follows:
     $R := \{\mathsf{id} \in \{0,1\}^n \mid \exists \mathsf{upk}, \mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}] = (\mathsf{id}, \mathtt{revoked}, -)\} \cup \{\mathsf{id}_{\mathsf{upk}}\}$;
  3. Computes $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}) \leftarrow \mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$.
  4. Queries $\langle t+1, \mathsf{acc}\rangle$ and receives σ, where t is included in $\mathsf{info}^{\mathsf{ipk}}_t$.

  Finally, it updates $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}] \leftarrow (\mathsf{id}_{\mathsf{upk}}, \mathtt{revoked}, -)$ and returns $\mathsf{info}^{\mathsf{ipk}}_{t+1} := (t+1, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$.
- For each query to $\mathsf{RtrKIReg}(\mathsf{ipk})$, $\mathcal{B}_{1,1}$ returns $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$ and terminates, $\mathcal{B}_2$ computes $(\mathsf{upk}, \mathsf{usk}, \mathsf{id}^*, \pi^*_{\mathsf{acc}}, \theta^*, r_1, r_2) \leftarrow \mathsf{SimExt}^1_1(\mathsf{crs}_1, \xi, \mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle, \pi)$, where $(C, \tau, \pi) \leftarrow \Sigma$. Then, if $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$ is empty, then it outputs $(\langle \mathsf{upk}, \mathsf{id}^*, \mathsf{item}\rangle, \theta^*)$. Else if $\mathsf{id}^* \neq \mathsf{id}_{\mathsf{upk}}$, it outputs $(\langle \mathsf{upk}, \mathsf{id}^*, \mathsf{item}\rangle, \theta^*)$ and terminates. Otherwise, it outputs $(\langle t, \mathsf{acc}\rangle, \sigma)$ and terminates, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$.

When the event $E \wedge \neg E_{\mathsf{bad}}$ occurs, there are two cases: (1) $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$ is empty or $\mathsf{id}^* \neq \mathsf{id}_{\mathsf{upk}}$, which means $\langle \mathsf{upk}, \mathsf{id}^*, \mathsf{item}\rangle$ has never been queried, or (2)

$\mathsf{id}^* = \mathsf{id}_{\mathsf{upk}}$ and acc included in the output of $\mathcal{A}$ is not included in the responses from queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$. In each case, $\mathcal{B}_2$ has a successful attack on the EUF-CMA security of SIG since we have $\mathsf{SIG.Ver}(\mathsf{vk}, \langle \mathsf{upk}, \mathsf{id}^*, \mathsf{item} \rangle, \theta^*) = 1$ (because the Verify algorithm returns 1) or $\mathsf{SIG.Ver}(\mathsf{vk}, \langle t, \mathsf{acc} \rangle, \sigma) = 1$, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$ without querying each message to the challenge oracle of the EUF-CMA game. Therefore, we have $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_{\mathsf{bad}}] \leq 2 \cdot \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG},\mathcal{B}_2}(n)$.

Next, we analyze the probability $\Pr[F]$. Let $(\mathsf{tpk}, \mathsf{tsk}) = ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$ be the honestly generated key pair for the tracer, ipk be the honest key issuer, and $(\mathsf{ipk}, \mathsf{item}, \mathsf{M}, \Sigma)$ be the output of $\mathcal{A}$ in the experiment. When the event F occurs, the Trace algorithm (1) outputs $\perp$, or (2) successfully decrypts $\mathsf{upk} \leftarrow \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$ but fails to make a valid proof of Π_2, where $(c_1, c_2) \leftarrow C$. As a matter of fact, the latter case never happens as long as Π_2 is correct. Since the Trace algorithm honestly runs, we have $\mathsf{upk} \leftarrow \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$ and $\Pi_{\mathsf{Trace}} \leftarrow \mathsf{NIZK.Prove}_2(\mathsf{crs}_2, \epsilon, \langle \mathsf{ek}_1, c_1, \mathsf{upk} \rangle, \mathsf{tsk})$, where $(C, \tau, \pi) \leftarrow \Sigma$, so we clearly have $(\langle \mathsf{ek}_1, c_1, \mathsf{upk} \rangle, \mathsf{tsk}) \in \rho_2$. In addition, when $\mathcal{A}$ wins, we have $\mathsf{NIZK.Verify}_1(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \pi) = 1$, where $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$. Thus, we always have $\mathsf{Judge}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}, \Pi_{\mathsf{Trace}}) = 1$ from its definition. Therefore, we only consider the former case.

When the former case occurs, i.e., $\mathsf{NIZK.Verify}_1(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \pi) = 1$ but Trace outputs $\perp$, $\mathcal{A}$ breaks the soundness of Π_1. More precisely, we have the following:

Lemma D.13. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[F] \leq \mathsf{Adv}^{\mathsf{sound}}_{\Pi_1,\mathcal{B}}(n). \tag{20}$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ which causes the event F. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the soundness of Π_1, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, given crs_1, randomly chooses $r_{\mathsf{PKE}} \leftarrow \{0,1\}^{\mathsf{poly}(n)}$, computes $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})$, and $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$, $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$, and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk}, \mathsf{tsk})$. It is easy to see that $\mathcal{B}$ can respond to each of queries from $\mathcal{A}$ in the same way as defined in Section B.1. When $\mathcal{A}$ outputs $(\mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$ and terminates, $\mathcal{B}$ outputs $\perp$ if $|\mathsf{HKIL}| \neq 1$. Otherwise, it outputs $(\mathsf{M}, \mathsf{X}, \pi)$ and terminates, where $(C, \tau, \pi) \leftarrow \Sigma$, $\mathsf{ipk} \leftarrow \mathsf{HKIL}$, and $\mathsf{X} \leftarrow \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle$.

The above completes the description of $\mathcal{B}$. As discussed above, we have $\perp \leftarrow \mathsf{Trace}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{tsk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma)$ when the event F occurs. From our construction of the Trace algorithm, the $\mathsf{PKE.Dec}$ algorithm fails, i.e., $\perp \leftarrow \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$. Thus, we have $\langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle \notin L_{\rho_1}$. On the other hand, we also have $\mathsf{Verify}(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma) = 1$. Therefore, $\mathcal{B}$ has a successful attack on the soundness of Π_1, i.e., $\Pr[F] \leq \mathsf{Adv}^{\mathsf{sound}}_{\Pi_1,\mathcal{B}}(n)$.

Theorem 5.4 now follows immediately from (18)-(20).

D.4 Proof of Theorem 5.5

Proof. Let $\mathcal{A}$ be a PPT adversary in the experiment $\mathsf{Exp}^{\mathsf{unf}}_{\mathsf{ARS},\mathcal{A}}(n)$ defined in Figure 5. Let $(\mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}_{\mathsf{Trace}}, \Pi_{\mathsf{Trace}})$ be the output by $\mathcal{A}$ in the experiment and t be included in info. When $\mathcal{A}$ wins, we have $t \notin \mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}_{\mathsf{Trace}}]$. Let E denote the event $\mathcal{A}$ wins the experiment $\mathsf{Exp}^{\mathsf{unf}}_{\mathsf{ARS},\mathcal{A}}(n)$. In addition, let E_{fail} be the event that $\mathcal{A}$ wins but the extractor Ext fails to extract the witness from the proof π included in Σ, and E_s be the event that $\mathsf{upk}_{\mathsf{Trace}}$ and upk included in the witness extracted from π are different. We clearly have $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{ARS},\mathcal{A}}(n) \leq \Pr[E \wedge E_{\mathsf{fail}}] + \Pr[E \wedge E_s \wedge \neg E_{\mathsf{fail}}] + \Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s]$. We will analyze the probability that each event occurs in order.

Lemma D.14. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[E \wedge E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}}(n). \tag{21}$$

Proof. We first modify the game slightly as follows. In the $\mathsf{RepSetup}$ algorithm, we compute $(\mathsf{crs}_1, \mathsf{td}_1, \xi) \leftarrow \mathsf{SimExt}^1_0(1^n)$ instead of $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$. Note that this affects nothing about the probability $\mathcal{A}$ wins.

We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event E_{fail}. Then, we can construct another PPT adversary $\mathcal{B}$ attacking the knowledge soundness of Π_1. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, given crs_1, computes $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$ and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp})$. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ in the same manner as defined in Section B.1. Note that $\mathcal{B}$ can respond correctly to all queries since it generates all parameters for ARS except crs_1, which is honestly generated by the challenger for the knowledge soundness of Π_1. When $\mathcal{A}$ outputs $(\mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}_{\mathsf{Trace}}, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}$ outputs $\perp$ if $|\mathsf{HKIL}| \neq 1$. Otherwise, it outputs $(\mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle, \pi)$ and terminates, where $\mathsf{ipk} \leftarrow \mathsf{HKIL}$ and $(C, \tau, \pi) \leftarrow \Sigma$.

The above completes the description of $\mathcal{B}$. When the event E_{fail} occurs, we have $\mathsf{NIZK.Verify}(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \pi) = 1$ (since $\mathcal{A}$ wins) and $(\mathsf{X}, \mathsf{W}) \notin \rho_1$, where $\mathsf{W} \leftarrow \mathsf{SimExt}^1_1(\mathsf{crs}_1, \mathsf{M}, \mathsf{X}, \pi)$ (since all extractor fail to extract a valid witness). In addition, $\mathcal{B}$ never queried to the zero-knowledge simulation oracle clearly, i.e., $L_{\mathcal{S}} = \emptyset$. Therefore, when the event E_{fail} occurs, $\mathcal{B}$ has a a successful attack on the simulation extractability of Π_1, i.e., $\Pr[E \wedge E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}}(n)$.

Lemma D.15. *There exists a PPT algorithm $\mathcal{B}$ such that*

$$\Pr[E \wedge E_s \wedge \neg E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{sound}}_{\Pi_2,\mathcal{B}}(n). \tag{22}$$

Proof. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E_s \wedge \neg E_{\mathsf{fail}}$. Then, we can construct another PPT adversary $\mathcal{B}$ attacking the soundness of Π_2. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, given crs_2, computes $(\mathsf{crs}_1, \mathsf{td}, \xi) \leftarrow \mathsf{SimExt}^1_0(1^n)$ and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp})$. $\mathcal{B}$ responds to each of queries from $\mathcal{A}$ in the same manner as defined in Section B.1. Note that $\mathcal{B}$ can respond correctly to all queries since it generates all parameters for ARS except crs_2, which is honestly generated by the challenger for the soundness of Π_2. When $\mathcal{A}$ outputs $(\mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}_{\mathsf{Trace}}, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}$ outputs $\perp$ if $|\mathsf{HKIL}| \neq 1$. Otherwise, it outputs $(\epsilon, \langle \mathsf{ek}_1, c_1, \mathsf{upk}_{\mathsf{Trace}} \rangle, \Pi_{\mathsf{Trace}})$ and terminates, where $(C, \tau, \pi) \leftarrow \Sigma$.

The above completes the description of $\mathcal{B}$. Let $\langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2 \rangle \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle)$, where $\mathsf{ipk} \leftarrow \mathsf{HKIL}$. When the event E_s occurs, we have $\mathsf{upk}_{\mathsf{Trace}} \neq \mathsf{upk}$. If the event E_{fail} never occurs, we always have $c_1 = \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1)$. Thus, when the event $E_s \wedge \neg E_{\mathsf{fail}}$ occurs, $\langle \mathsf{ek}_1, c_1, \mathsf{upk}_{\mathsf{Trace}} \rangle \notin L_{\rho_2}$. On the other hand, we have $\mathsf{NIZK.Verify}_2(\mathsf{crs}_2, \epsilon, \langle \mathsf{ek}_1, c_1, \mathsf{upk}_{\mathsf{Trace}} \rangle, \Pi_{\mathsf{Trace}}) = 1$ since $\mathcal{A}$ wins. Therefore, $\mathcal{B}$ has a successful attack on the soundness of Π_2, i.e., $\Pr[E \wedge E_s \wedge \neg E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{sound}}_{\Pi_2, \mathcal{B}}(n)$.

Lemma D.16. *There exist PPT algorithms $\mathcal{B}_1, \mathcal{B}_2$ such that*

$$\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s] \leq \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ACC}, \mathcal{B}_1}(n) + 2 \cdot \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG}, \mathcal{B}_2}. \tag{23}$$

Proof. Let $(\mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}_{\mathsf{Trace}}, \Pi_{\mathsf{Trace}})$ be the output of $\mathcal{A}$ in the experiment. Since the event E_{fail} never occurs, we can extract the witness $\langle \mathsf{upk}, \mathsf{usk}, \mathsf{id}^*, \mathsf{wit}^*_{\mathsf{id}}, \theta^*, r_1, r_2 \rangle$ from π included in Σ. We divide the case into two depending on whether the three conditions are satisfied: (a) the extracted upk is included in the queries from $\mathcal{A}$ to the $\mathsf{SndToKI}$ oracle, (b) the extracted id^* equals to $\mathsf{id}_{\mathsf{upk}}$ that is recorded in $\mathsf{reg}^{\mathsf{ipk}}_{\mathsf{ki}}[\mathsf{upk}]$, and (c) $\mathcal{A}$ outputs $\mathsf{info} = (t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma)$ such that it obtained $\langle t, \mathsf{acc} \rangle$ by querying to the $\mathsf{SndToKI}$ or $\mathsf{RevUser}$ oracle. Roughly, in case the above conditions are satisfied, $\mathcal{A}$ breaks the soundness of ACC. Otherwise, $\mathcal{A}$ breaks the EUF-CMA security of SIG. We denote the former case by the event E_{bad}. We clearly have $\Pr[E \wedge \neg E_{\mathsf{fail}} \neg E_s] \leq \Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge E_{\mathsf{bad}}] + \Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge \neg E_{\mathsf{bad}}]$. We analyze each probability in order.

Firstly, we show $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge E_{\mathsf{bad}}] \leq \mathsf{poly}(n) \cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ACC}, \mathcal{B}_1}$. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge E_{\mathsf{bad}}$. Then, we can construct another PPT adversary $\mathcal{B}_1$ attacking the soundness of ACC. The description of $\mathcal{B}_1$ is as follows.

$\mathcal{B}_{1,1}$, given $\mathsf{pp}_{\mathsf{acc}}$, computes $(\mathsf{crs}_1, \mathsf{td}, \xi) \leftarrow \mathsf{SimExt}_0^1(1^n)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$, $((\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$, and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Let the number of queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$ from $\mathcal{A}$ be $K = \mathsf{poly}(n)$. $\mathcal{B}_{1,1}$ randomly chooses $k \leftarrow [K]$ and outputs the k-th set of revoked users R_k to the challenger to the soundness game. Then, $\mathcal{B}_{1,1}$ runs $\mathcal{A}(\mathsf{pp})$. $\mathcal{B}_{1,1}$ responds to each of queries from $\mathcal{A}$ in the same way as the proof of Lemma D.12. Then, $\mathcal{B}_{1,2}$ receives $(\mathsf{st}, \mathsf{acc}_k, \mathsf{aux}_{\mathsf{acc}})$ from its challenger and continues to run $\mathcal{A}$.

When $\mathcal{A}$ outputs $(\mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}_{\mathsf{Trace}}, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}_{1,2}$ outputs $\perp$ and terminates if $\langle t, \mathsf{acc} \rangle \neq \langle t_k, \mathsf{acc}_k \rangle$, where acc is included in info. Otherwise, it computes $(\mathsf{upk}, \mathsf{usk}, \mathsf{id}^*, \pi^*_{\mathsf{acc}}, \theta, r_1, r_2) \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau \rangle, \pi)$, where $(C, \tau, \pi) \leftarrow \Sigma$, and outputs $(\mathsf{id}^*, \mathsf{wit}^*_{\mathsf{id}})$ and terminates.

The above completes the description of $\mathcal{B}_{1,2}$. Since the event E_s never occurs, we always have $\mathsf{upk}_{\mathsf{Trace}} = \mathsf{upk}$. When the event $E \wedge E_{\mathsf{bad}}$ occurs, we have (1) $\mathsf{id}^* = \mathsf{id}_{\mathsf{upk}}$, (2) acc included in the output of $\mathcal{A}$ is also included in one of responses from queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$, and (3) $t \notin \mathsf{IsActive}[\mathsf{ipk}][\mathsf{item}][\mathsf{upk}]$, which means

$\mathsf{id}_{\mathsf{upk}} \in R$, where $(\mathsf{id}_{\mathsf{upk}}, -, -) \leftarrow \mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$. The probability that $\mathsf{acc} = \mathsf{acc}_k$ is $1/K$. In case $\mathsf{acc} = \mathsf{acc}_k$, we have $\mathsf{ACC.Verify}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{acc}_k, \mathsf{id}^*, \pi^*) = 1$, where $\pi^* \leftarrow \mathsf{ACC.Prove}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{aux}_{\mathsf{acc}}, \mathsf{id}^*, \mathsf{wit}_{\mathsf{id}}^*)$, since the Verify algorithm returns 1. Therefore, $\mathcal{B}$ has a a successful attack on the soundness of ACC with probability $1/K$, i.e., $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge E_{\mathsf{bad}}] \leq K \cdot \mathsf{Adv}_{\mathsf{ACC},\mathcal{B}}^{\mathsf{sound}}(n)$.

Secondly, we show $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge \neg E_{\mathsf{bad}}] \leq 2 \cdot \mathsf{Adv}_{\mathsf{SIG}\mathcal{B}_2}^{\mathsf{unf}}(n)$. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E_1 \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge \neg E_{\mathsf{bad}}$. Then, we can construct another PPT adversary $\mathcal{B}_2$ attacking the EUF-CMA security of SIG. The description of $\mathcal{B}_2$ is as follows.

$\mathcal{B}_2$, given vk, randomly chooses $r_{\mathsf{PKE}} \leftarrow \{0,1\}^{\mathsf{poly}(n)}$, computes $(\mathsf{crs}_1, \mathsf{td}, \xi) \leftarrow \mathsf{SimExt}_1^1(1^n)$, $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$, $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})$, and $(\mathsf{ek}_2, \mathsf{dk}_2) \leftarrow \mathsf{PKE.KG}(1^n)$, and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$, $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow ((\mathsf{ek}_1, \mathsf{ek}_2), (\mathsf{dk}_1, \mathsf{dk}_2, r_{\mathsf{PKE}}))$, and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Then, $\mathcal{B}_2$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{tpk}, \mathsf{tsk})$. $\mathcal{B}_2$ responds to each of queries from $\mathcal{A}$ in the same way as the proof of Lemma D.12. When $\mathcal{A}$ outputs $(\mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \mathsf{upk}_{\mathsf{Trace}}, \Pi_{\mathsf{Trace}})$ and terminates, $\mathcal{B}_2$ computes $(\mathsf{upk}, \mathsf{usk}, \mathsf{id}^*, \pi_{\mathsf{acc}}^*, \theta^*, r_1, r_2) \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \langle C, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau\rangle, \pi)$, where $(C, \tau, \pi) \leftarrow \Sigma$. Then, if $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$ is empty or $\mathsf{id}^* \neq \mathsf{id}_{\mathsf{upk}}$, it outputs $(\langle \mathsf{upk}, \mathsf{id}^*, \mathsf{item}\rangle, \theta^*)$ and terminates. Otherwise, it outputs $(\langle t, \mathsf{acc}\rangle, \sigma)$ and terminates, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$.

When the event $E \wedge \neg E_{\mathsf{bad}}$ occurs, there are two cases: (1) $\mathsf{reg}_{\mathsf{ki}}^{\mathsf{ipk}}[\mathsf{upk}]$ is empty or $\mathsf{id}^* \neq \mathsf{id}_{\mathsf{upk}}$, which means $\langle \mathsf{upk}, \mathsf{id}^*, \mathsf{item}\rangle$ has never been queried, or (2) $\mathsf{id}^* = \mathsf{id}_{\mathsf{upk}}$ and acc included in the output of $\mathcal{A}$ is not included in the responses from queries to $\mathsf{SndToKI}$ and $\mathsf{RevUser}$. In the former case, $\mathcal{B}_2$ has a successful attack on the EUF-CMA security of SIG since we have $\mathsf{SIG.Ver}(\mathsf{vk}, \langle \mathsf{upk}, \mathsf{id}^*, \mathsf{item}\rangle, \theta^*) = 1$ (because the Verify algorithm returns 1). In the latter case, $\mathcal{B}_2$ also has a successful attack on the EUF-CMA security of SIG since we have $\mathsf{SIG.Ver}(\mathsf{vk}, \langle t, \mathsf{acc}\rangle, \sigma) = 1$, where $(t, \mathsf{acc}, \mathsf{aux}_{\mathsf{acc}}, \sigma) \leftarrow \mathsf{info}$. Therefore, $\mathcal{B}_2$ has a a successful attack on the EUF-CMA security of SIG, i.e., $\Pr[E \wedge \neg E_{\mathsf{fail}} \wedge \neg E_s \wedge \neg E_{\mathsf{bad}}] \leq 2 \cdot \mathsf{Adv}_{\mathsf{SIG},\mathcal{B}_2}^{\mathsf{unf}}(n)$.

Theorem 5.5 now follows immediately from (21)-(23).

D.5 Proof of Theorem 5.6

Proof. We assume that there exists a PPT adversary $\mathcal{A}$ that attacks the tracing soundness of ARS. Then, we can construct another PPT adversary $\mathcal{B}$ attacking the soundness of Π_2. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, given crs_2, computes $\mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}_1(1^n)$ and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$ and $\mathsf{HKIL}, \mathsf{IsActive} = \emptyset$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp})$. When $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{M}, \Sigma, \{\mathsf{upk}_i, \Pi_{\mathsf{Trace},i}\}_{i=0,1})$ and terminates, $\mathcal{B}$ randomly chooses $b \in \{0,1\}$ and outputs $(\epsilon, \langle \mathsf{ek}_1, c_1, \mathsf{upk}_b\rangle, \Pi_{\mathsf{Trace},b})$ and terminates, where $(C, \tau, \pi) \leftarrow \Sigma$.

The above completes the description of $\mathcal{B}$. When $\mathcal{A}$ wins, we have $\mathsf{NIZK.Verify}_2(\mathsf{crs}_2, \epsilon, \langle \mathsf{ek}_1, c_1, \mathsf{upk}_i\rangle, \Pi_{\mathsf{Trace},i}) = 1$ for $i = 0, 1$ and $\mathsf{upk}_0 \neq \mathsf{upk}_1$, which means that one of the followings hold: $\mathsf{upk}_0 = \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$ or $\mathsf{upk}_1 =$

$\mathsf{PKE.Dec}(\mathsf{dk}_1, c_1)$. Thus, we have $\langle \mathsf{ek}_1, c_1, \mathsf{upk}_i \rangle \notin L_{\rho_2}$ for $i = 0$ or 1. Therefore, $\mathcal{B}$ has a successful attack on the soundness of Π_2 with probability $1/2$, i.e., $\mathsf{Adv}^{\mathsf{trace\text{-}sound}}_{\mathsf{ARS},\mathcal{A}}(n) \leq 2 \cdot \mathsf{Adv}^{\mathsf{sound}}_{\Pi_2,\mathcal{B}}(n)$.

D.6 Proof of Theorem 5.7

Proof. Let us fix a PPT adversary $\mathcal{A}$ attacking the public-linkability of ARS and the value of the security parameter n. The attack game used to define the public-linkability is defined in Figure 7. Let E be the event that $\mathcal{A}$ wins in the original attack game, $(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{upk}, \{\mathsf{M}_i, \Sigma_i, \Pi_{\mathsf{Trace},i}\}_{i=0,1})$ be the output of $\mathcal{A}$, E_{fail} be the event that the knowledge extractor Ext fails to extract a witness W_i from π_i included in Σ_i for $i = 0$ or 1, and E_{bad} be the event that $\mathsf{upk} \neq \mathsf{upk}_0$ or $\mathsf{upk} \neq \mathsf{upk}_1$, where upk_i is included in W_i for $i = 0, 1$. We clearly have $\Pr[E] \leq \Pr[E \wedge E_{\mathsf{fail}}] + \Pr[E \wedge E_{\mathsf{bad}} \wedge \neg E_{\mathsf{fail}}] + \Pr[E \wedge \neg E_{\mathsf{bad}} \wedge \neg E_{\mathsf{fail}}]$. We will analyze each probability in order.

Beforehand, we slightly modify the game as follows: we use $\mathsf{SimExt}_1 = (\mathsf{SimExt}^1_0, \mathsf{SimExt}^1_1)$ instead of $\mathsf{NIZK.Setup}_1$. We note that this has negligible effect on the probability $\mathcal{A}$ wins as seen in the proof of appendix D.2.

Lemma D.17. *There exist PPT adversaries $\mathcal{B}$ such that*

$$\Pr[E \wedge E_{\mathsf{fail}}] \leq 2 \cdot \mathsf{Adv}^{\mathsf{se}}_{\Pi_1,\mathcal{B}}(n). \tag{24}$$

Proof. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E \wedge E_{\mathsf{fail}}$. Then, we can construct another PPT adversary $\mathcal{B}$ attacking the simulation extractability of Π_1. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, given crs_1, computes $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n)$ and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp})$. When $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{upk}, \{\mathsf{M}_i, \Sigma_i, \Pi_{\mathsf{Trace},i}\}_{i=0,1})$ and terminates, $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$ and outputs $(\mathsf{M}_b, \langle C_b, \mathsf{ipk}, \mathsf{tpk}, t, \mathsf{item}, \tau_b \rangle, \pi_b)$ and terminates, where $(C_b, \tau_b, \pi_b) \leftarrow \Sigma_b$.

The above completes the description of $\mathcal{B}$. When the event E_{fail} occurs, for $i = 0$ or 1, we have $(\mathsf{X}_i, \mathsf{W}_i) \notin \rho_1$, where $\mathsf{X}_i \leftarrow \langle C_i, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau_i \rangle$ and $\mathsf{W}_i \leftarrow \mathsf{SimExt}^1_1(\mathsf{crs}_1, \xi, \mathsf{M}_i, \mathsf{X}_i, \pi_i)$. On the other hand, when the event E occurs (i.e., $\mathcal{A}$ wins), for both $i = 0$ and 1, we have $\mathsf{NIZK.Verify}_1(\mathsf{crs}_1, \mathsf{M}_i, \mathsf{X}_i, \pi_i) = 1$. Therefore, $\mathcal{B}$ has a successful attack on the simulation extractability of Π_1 with probability $1/2$ (since it never queries to the zero-knowledge simulation oracle), i.e., $\Pr[E \wedge E_{\mathsf{fail}}] \leq 2 \cdot \mathsf{Adv}^{\mathsf{ks}}_{\Pi_1,\mathcal{B}}(n)$.

Lemma D.18. *There exists a PPT adversary $\mathcal{B}$ such that*

$$\Pr[E \wedge E_{\mathsf{bad}} \wedge \neg E_{\mathsf{fail}}] \leq 2 \cdot \mathsf{Adv}^{\mathsf{sound}}_{\Pi_2,\mathcal{B}}(n). \tag{25}$$

Proof. We assume that there exists a PPT adversary $\mathcal{A}$ that causes the event $E \wedge E_{\mathsf{bad}} \wedge \neg E_{\mathsf{fail}}$. Then, we can construct another PPT adversary $\mathcal{B}$ attacking the soundness of Π_2. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$, given crs_2, computes $(\mathsf{crs}_1, \mathsf{td}, \xi) \leftarrow \mathsf{SimExt}^1_0(1^n)$ and sets $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp})$. When $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{upk}, \{\mathsf{M}_i, \Sigma_i, \Pi_{\mathsf{Trace},i}\}_{i=0,1})$

and terminates, $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$ and outputs $(\epsilon, \langle \mathsf{tpk}, c_{1,b}, \mathsf{upk}\rangle, \pi_b)$ and terminates, where $(C_b, \tau_b, \pi_b) \leftarrow \Sigma_b$ and $(c_{1,b}, c_{2,b}) \leftarrow C_b$.

The above completes the description of $\mathcal{B}$. Let $\langle \mathsf{upk}_b, \mathsf{usk}_b, \mathsf{id}_b, \mathsf{wit}_{\mathsf{id}_b}, \theta_b, r_{1,b}, r_{2,b}\rangle \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}, \langle C_b, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau_b\rangle, \pi)$. When the event $E \wedge \neg E_{\mathsf{fail}}$ occurs, we have $(\mathsf{X}_i, \mathsf{W}_i) \in \rho_1$, where $\mathsf{X}_i \leftarrow \langle C_i, \mathsf{ipk}, \mathsf{tpk}, \mathsf{acc}, \mathsf{item}, \tau_i\rangle$ and $\mathsf{W}_i \leftarrow \langle \mathsf{upk}_i, \mathsf{usk}_i, \mathsf{id}_i, \mathsf{wit}_{\mathsf{id}_i}, \theta_i, r_{1,i}, r_{2,i}\rangle$, for both $i = 0$ and 1. Thus, we have $c_i = \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}_i; r_{1,i})$ for both $i = 0$ and 1. In addition, when the event E_{bad} occurs, we have $\mathsf{upk} \neq \mathsf{upk}_0$ or $\mathsf{upk} \neq \mathsf{upk}_1$. On the other hand, when the event E occurs (i.e., $\mathcal{A}$ wins), we have $\mathsf{NIZK.Verify}_2(\mathsf{crs}_2, \epsilon, \langle \mathsf{tpk}, c_i, \mathsf{upk}\rangle, \Pi_{\mathsf{Trace},i}) = 1$ for both $i = 0$ and 1. Thus, if $\mathsf{upk} \neq \mathsf{upk}_b$, then we have $\langle \mathsf{tpk}, c_b, \mathsf{upk}\rangle \notin L_{\rho_2}$. Therefore, since b is randomly chosen, $\mathcal{B}$ has a successful attack on the soundness of Π_2 with probability 1/2, i.e., $\Pr[E \wedge E_{\mathsf{bad}} \wedge \neg E_{\mathsf{fail}}] \leq 2 \cdot \mathsf{Adv}^{\mathsf{sound}}_{\Pi_2,\mathcal{B}}(n)$.

Lemma D.19. *There exists a PPT adversary $\mathcal{B}$ such that*

$$\Pr[E \wedge \neg E_{\mathsf{bad}} \wedge \neg E_{\mathsf{fail}}] \leq \mathsf{Adv}^{\mathsf{link}}_{\mathsf{LIT},\mathcal{B}}(n). \tag{26}$$

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ attacking the public linkability of ARS. Then, we can construct another PPT adversary $\mathcal{B}$ that breaks the linkability of LIT whose success probability is the same as that of $\mathcal{A}$, which implies the lemma. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ computes $(\mathsf{crs}_1, \mathsf{td}, \xi) \leftarrow \mathsf{SimExt}_0^1(1^n)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK.Setup}_2(1^n))$, and set $\mathsf{pp} \leftarrow (\mathsf{crs}_1, \mathsf{crs}_2)$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp})$. Consider that $\mathcal{A}$ outputs $(\mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \mathsf{upk}, \{\mathsf{M}_i, \Sigma_i, \Pi_{\mathsf{Trace},i}\}_{i=0,1})$ and terminates. For $i = 0, 1$, $\mathcal{B}$ parses $(C_i, \tau_i, \pi_i) \leftarrow \Sigma_i$ and computes $\langle \mathsf{upk}_i, \mathsf{usk}_i, \mathsf{id}_i, \mathsf{wit}_{\mathsf{id}_i}, \theta_i, r_{1,i}, r_{2,i}\rangle \leftarrow \mathsf{SimExt}_1^1(\mathsf{crs}_1, \xi, \mathsf{M}_i, \mathsf{X}_i, \pi_i)$, where $\mathsf{X}_i \leftarrow \langle C_i, \mathsf{ipk}, \mathsf{tpk}, \mathsf{info}, \mathsf{item}, \tau_i\rangle$. Finally, $\mathcal{B}$ outputs $(\mathsf{upk}, \mathsf{usk}_0, \mathsf{usk}_1, \langle \mathsf{ipk}, \mathsf{item}\rangle, \tau_0, \tau_1)$ and terminates.

The above completes the description of $\mathcal{B}$. If the event E_{bad} never occurs, we always have $\mathsf{upk} = \mathsf{upk}_0 = \mathsf{upk}_1$. If the event E_{fail} never occurs, we always have $\mathsf{LIT.ChkTag}(\mathsf{upk}_i, \mathsf{usk}_i, \langle \mathsf{ipk}, \mathsf{item}\rangle, \tau_i) = 1$ for $i = 0, 1$ since $(\mathsf{X}_i, \mathsf{W}_i) \in \rho_1$, where $\mathsf{W}_i \leftarrow \langle \mathsf{upk}_i, \mathsf{usk}_i, \mathsf{id}_i, \mathsf{wit}_{\mathsf{id}_i}, \theta_i, r_{1,i}, r_{2,i}\rangle$. When the event E occurs (i.e., $\mathcal{A}$ wins), we have $\mathsf{LIT.Link}(\tau_0, \tau_1) = 0$. Therefore, $\mathcal{B}$ has a successful attack on the linkability of LIT, i.e., $\mathsf{Adv}^{\mathsf{public\text{-}link}}_{\mathsf{ARS},\mathcal{A}}(n) \leq \mathsf{Adv}^{\mathsf{link}}_{\mathsf{LIT},\mathcal{B}}(n)$.

Theorem 5.7 now follows immediately from (24)-(26).

E Instantiation of Our Generic Construction

In this section, we provide a pairing-based instantiation of our generic construction of anonymous reputation system. Below, we provide an instantiation of each building blocks from pairing.

Public-Key Encryption. We use the ElGamal encryption scheme [21] from the decision Diffie-Hellman assumption.

Signatures. We use the Abe-Groth-Haralambiev-Ohkubo structure-preserving signature scheme [1]. We need two types of schemes, one with message space $\mathbb{G}_1^{\ell_{\mathrm{sig},1}}$ and one with message space $\mathbb{G}_2^{\ell_{\mathrm{sig},2}}$, which we present below.

The description of the scheme with message space $\mathbb{G}_1^{\ell_{\mathrm{sig},1}}$ is below.

$\mathsf{SIG.KG}(1^n)$**.** Compute $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^n)$. Choose $u_1, \ldots, u_{\ell_{\mathrm{sig},1}}$, and $v \leftarrow \mathbb{Z}_p^*$ and compute $U_1 \leftarrow h^{u_1}, \ldots, U_{\ell_{\mathrm{sig},1}} \leftarrow h^{u_{\ell_{\mathrm{sig},1}}}$, and $V \leftarrow g^v$. Set $\mathsf{vk} \leftarrow (U_1, \ldots, U_{\ell_{\mathrm{sig},1}}, V)$ and $\mathsf{sk} \leftarrow (u_1, \ldots, u_{\ell_{\mathrm{sig},1}}, v)$ and output $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{SIG.Sign}(\mathsf{sk}, (m_1, \ldots, m_{\ell_{\mathrm{sig},1}}))$**.** Choose $r \leftarrow \mathbb{Z}_p^*$ and set $R \leftarrow h^r$, $S \leftarrow R^v$, and $T \leftarrow (g \prod_{i=1}^{\ell_{\mathrm{sig},1}} m_i^{-u_i})^{1/r}$. Output $\sigma \leftarrow (R, S, T)$.

$\mathsf{SIG.Rerand}(\mathsf{vk}, (R, S, T))$**.** Choose $s \leftarrow \mathbb{Z}_p^*$ and compute $(\tilde{R}, \tilde{S}, \tilde{T}) \leftarrow (R^s, S^s, T^{1/s})$. Output $(\tilde{R}, \tilde{S}, \tilde{T})$.

$\mathsf{SIG.Ver}(\mathsf{vk}, (m_1, \ldots, m_{\ell_{\mathrm{sig},1}}), (R, S, T))$**.** Verify $T, m_1, \ldots, m_{\ell_{\mathrm{sig},1}} \in \mathbb{G}_1$, $R, S \in \mathbb{G}_2$, $e(V, R) = e(g, S)$, and $e(T, R) \prod_{i=1}^{\ell_{\mathrm{sig},1}} e(m_i, U_i) = e(g, h)$. If all the equations hold, output 1. Otherwise, output 0.

Theorem E.1. *The above scheme is EUF-CMA secure in the generic bilinear group model.*

The description of the scheme with message space $\mathbb{G}_2^{\ell_{\mathrm{sig},2}}$ is below.

$\mathsf{SIG.KG}'(1^n)$**.** Compute $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^n)$. Choose $u'_1, \ldots, u'_{\ell_{\mathrm{sig},2}}$, and $v' \leftarrow \mathbb{Z}_p^*$ and compute $U'_1 \leftarrow g^{u'_1}, \ldots, U'_{\ell_{\mathrm{sig},2}} \leftarrow g^{u'_{\ell_{\mathrm{sig},2}}}$, and $V' \leftarrow h^{v'}$. Set $\mathsf{vk} \leftarrow (U'_1, \ldots, U'_{\ell_{\mathrm{sig},2}}, V')$ and $\mathsf{sk} \leftarrow (u'_1, \ldots, u'_{\ell_{\mathrm{sig},2}}, v')$ and output $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{SIG.Sign}'(\mathsf{sk}, (m'_1, \ldots, m'_{\ell_{\mathrm{sig},2}}))$**.** Choose $r' \leftarrow \mathbb{Z}_p^*$ and set $R' \leftarrow g^{r'}$, $S' \leftarrow (R')^{v'}$, and $T' \leftarrow (h \prod_{i=1}^{\ell_{\mathrm{sig},2}} (m'_i)^{-u'_i})^{1/r'}$. Output $\sigma \leftarrow (R', S', T')$.

$\mathsf{SIG.Rerand}'(\mathsf{vk}, (R', S', T'))$**.** Choose $s \leftarrow \mathbb{Z}_p^*$ and compute $(\tilde{R}', \tilde{S}', \tilde{T}') \leftarrow ((R')^s, (S')^s, (T')^{1/s})$. Output $(\tilde{R}', \tilde{S}', \tilde{T}')$.

$\mathsf{SIG.Ver}'(\mathsf{vk}, (m_1, \ldots, m_{\ell_{\mathrm{sig},2}}), (R, S, T))$**.** Verify $R', S' \in \mathbb{G}_1, T', m'_1, \ldots, m'_{\ell_{\mathrm{sig},2}} \in \mathbb{G}_2$, $e(R', V') = e(S', h)$, and $e(R', T') \prod_{i=1}^{\ell_{\mathrm{sig},2}} e(U'_i, m'_i) = e(g, h)$. If all the equations hold, output 1. Otherwise, output 0.

Theorem E.2. *The above construction is EUF-CMA secure in the generic bilinear group model.*

Linkable Indistinguishable Tag. We provide a simple construction of linkable indistinguishable tags from the DDH assumption. Here, $H \colon \{0,1\}^* \to \mathbb{G}_2$ is a cryptographic hash function modeled as a random oracle.

$\mathsf{LIT.KG}(1^n)$**.** Compute $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^n)$ and choose $x \leftarrow \mathbb{Z}_p$. Set $\mathsf{tagpk} \leftarrow g^x$ and $\mathsf{tagsk} \leftarrow x$ and output $(\mathsf{tagpk}, \mathsf{tagsk})$.

$\mathsf{LIT.Tag}(\mathsf{tagsk}, I)$**.** Compute $\tau \leftarrow H(I)^{\mathsf{tagsk}}$ and output τ.

$\mathsf{LIT.Link}(\tau_0, \tau_1)$**.** Check $\tau_0 = \tau_1$. If it holds, output 1. Otherwise, output 0.

$\mathsf{LIT.ChkKey}(\mathsf{tagpk}, \mathsf{tagsk})$**.** Check $\mathsf{tagpk} = g^{\mathsf{tagsk}}$. If it holds, output 1. Otherwise, output 0.

$\mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}, I, \tau)$**.** Check $\mathsf{tagpk} = g^{\mathsf{tagsk}}$ and $\tau = H(I)^{\mathsf{tagsk}}$. If they hold, output 1. Otherwise, output 0.

Theorem E.3. *If the DDH assumption on $\mathbb{G}_1$ for $\mathcal{G}$ holds, the above construction of linkable indistinguishable tags is indistinguishable in the random oracle model.*

Theorem E.4. *The above construction is linkable.*

Theorem E.5. *If the DL assumption on $\mathbb{G}_1$ for $\mathcal{G}$ holds, the above construction is key secret.*

Theorem E.6. *The above construction is key robust.*

In the following, we give proof of security of the DDH-based Instantiation of LIT.

Proof of Theorem E.3

Proof. Let $\mathcal{A}$ be a PPT adversary against the indistinguishability of the scheme. Let fix a security parameter n. We consider the following games. In the games, we assume that the challenge query I^* should be queried to the random oracle beforehand and any tag generation oracle query should be so.

Game_1**.** This is the real security game of the indistinguishability.
Game_2**.** In this game, each tag generation query of the form $(0, I)$ is responded with a fresh random element in $\mathbb{G}_1$. In addition, if $b = 0$, the challenge tag τ^* is set to be a fresh random element in $\mathbb{G}_1$.
Game_3**.** In this game, each tag generation query of the form $(1, I)$ is also responded with a fresh random element in $\mathbb{G}_1$. In addition, if $b = 0$, the challenge tag τ^* is also set to be a fresh random element in $\mathbb{G}_1$.

Let W_i be the event taht $\mathcal{A}$'s guess b' is equal to b in Game_i.
We bound the difference between the games one by one.

Lemma E.1. *If the DDH assumption on $\mathbb{G}_1$ holds for $\mathcal{G}$, there exists a PPT adversary $\mathcal{B}_1$ against the DDH assumption satisfying* $|\Pr[W_1] - \Pr[W_2]| = \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{G},\mathcal{B}_1}$.

Proof. We construct $\mathcal{B}_1$ as follows. Given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ and a DDH problem $(g^\alpha, g^\beta, g^{\alpha\beta+\gamma})$ where $\gamma = 0$ or a random element in $\mathbb{Z}_p \setminus \{0\}$, $\mathcal{B}_1$ sets $\mathsf{tagpk}_0 \leftarrow g^\alpha$, choose $\mathsf{tagsk}_1 \leftarrow \mathbb{Z}_p$, and sets $\mathsf{tagpk}_1 \leftarrow g^{\mathsf{tagsk}_1}$; then $\mathcal{B}_1$ runs $\mathcal{A}(\mathsf{tagpk}_0, \mathsf{tagpk}_1)$. When $\mathcal{A}$ issues a random oracle query I, $\mathcal{B}_1$ chooses u_I, $v_I \leftarrow \mathbb{Z}_p$, records u_I and v_I, and returns $g^{u_I}(g^\beta)^{v_I}$. When $\mathcal{A}$ issues a tag generation query $(0, I)$, $\mathcal{B}_1$ retrieves u_I and v_I and returns $(g^\alpha)^{u_I}(g^{\alpha\beta+\gamma})^{v_I}$. When $\mathcal{A}$ issues a tag generation query $(1, I)$, $\mathcal{B}_1$ retrieves u_I and v_I and returns $(g^{u_I}(g^\beta)^{v_I})^{\mathsf{tagsk}_1}$. When $\mathcal{A}$ issues a challenge query I^*, $\mathcal{B}_1$ chooses $b \leftarrow \{0, 1\}$. Then $\mathcal{B}_1$ retrieves u_{I^*} and v_{I^*}, returns $(g^\alpha)^{u_{I^*}}(g^{\alpha\beta+\gamma})^{v_{I^*}}$ if $b = 0$, and returns $(g^{u_{I^*}}(g^\beta)^{v_{I^*}})^{\mathsf{tagsk}_1}$ if $b = 1$. Once $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}_1$ outputs 1 if $b = b'$ and outputs 0 if $b \neq b'$.

We claim that if $\gamma = 0$ then $\mathcal{B}_1$ perfectly simulates Game_1 and that if $\gamma \neq 0$ then $\mathcal{B}_1$ perfectly simulates Game_2. To see this, we examine the joint distribution of the response $H(I)$ to the random oracle query I and the response τ to the tag generation query $(0, I)$. These values are computed as $H(I) = g^{u_I}(g^\beta)^{v_I}$ and $\tau = (g^\alpha)^{u_I}(g^{\alpha\beta+\gamma})^{v_I}$. Due to the randomness of u_I, $H(I)$ is distributed unifromly over $\mathbb{G}_1$. Furthermore, if $\gamma = 0$, we have that $\tau = (g^\alpha)^{u_I}(g^{\alpha\beta})^{v_I} = (g^{u_I+\beta v_I})^\alpha$. This means that τ is generated honestly using $\mathsf{tagsk}_0 = \alpha$. If $\gamma \neq 0$, due to the randomness of v_I, τ is distributed uniformly over $\mathbb{G}_1$ and independently of $H(I)$. Therefore, if $\gamma = 0$, $\mathcal{B}_1$ perfectly simulates Game_1 and if $\gamma \neq 0$, $\mathcal{B}_1$ perfectly simulates Game_2. Hence the lemma holds.

Lemma E.2. *If the DDH assumption on $\mathbb{G}_1$ holds for $\mathcal{G}$, there exists a PPT adversary $\mathcal{B}_2$ against the DDH assumption satisfying* $|\Pr[W_1] - \Pr[W_2]| = \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{G},\mathcal{B}_2}$.

Proof. We construct $\mathcal{B}_2$ as follows. Given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ and a DDH problem $(g^\alpha, g^\beta, g^{\alpha\beta+\gamma})$ where $\gamma = 0$ or a random element in $\mathbb{Z}_p \setminus \{0\}$, $\mathcal{B}_2$ chooses $\mathsf{tagpk}_0 \leftarrow \mathbb{G}_1$, sets $\mathsf{tagpk}_1 \leftarrow g^\alpha$, and runs $\mathcal{A}(\mathsf{tagpk}_0, \mathsf{tagpk}_1)$. When $\mathcal{A}$ issues a random oracle query I, $\mathcal{B}_2$ chooses u_I, $v_I \leftarrow \mathbb{Z}_p$, records u_I and v_I, and returns $g^{u_I}(g^\beta)^{v_I}$. When $\mathcal{A}$ issues a tag generation query $(0, I)$, $\mathcal{B}_2$ chooses a uniformly random element in $\mathbb{G}_1$ and returns it. When $\mathcal{A}$ issues a tag generation query $(1, I)$, $\mathcal{B}_2$ retrieves u_I and v_I and returns $(g^\alpha)^{u_I}(g^{\alpha\beta+\gamma})^{v_I}$. When $\mathcal{A}$ issues a challenge query I^*, $\mathcal{B}_2$ chooses $b \leftarrow \{0, 1\}$. If $b = 0$, then $\mathcal{B}_2$ chooses a uniformly random element in $\mathbb{G}_1$ and returns it. If $b = 1$, then $\mathcal{B}_2$ retrieves u_{I^*} and v_{I^*} and returns $(g^\alpha)^{u_{I^*}}(g^{\alpha\beta+\gamma})^{v_{I^*}}$. Once $\mathcal{A}$ outputs b' and terminates, $\mathcal{B}_2$ outputs 1 if $b = b'$ and outputs 0 if $b \neq b'$.

From a similar argument as in the above lemma, we have that if $\gamma = 0$, $\mathcal{B}_2$ perfectly simulates Game_2 and that if $\gamma \neq 0$, $\mathcal{B}_2$ perfectly simulates Game_3. Hence the lemma holds.

Using the lemmas above, we have that

$$\begin{aligned}
&\left|\Pr[W_1] - \frac{1}{2}\right| \\
&\qquad \leq |\Pr[W_1] - \Pr[W_2]| + |\Pr[W_2] - \Pr[W_3]| + \left|\Pr[W_3] - \frac{1}{2}\right| \\
&\qquad = \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{G},\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{G},\mathcal{B}_2}.
\end{aligned}$$

The last equality is due to the lemmas and the fact that in Game_3, the challenge bit is independent of the view of $\mathcal{A}$. This completes the proof.

Proof of Theorem E.4

Proof. Let $\mathcal{A}$ be an PPT adversary against the linkability of the scheme. Let $(\mathsf{tagpk}, \mathsf{tagsk}_0, \mathsf{tagsk}_1, I, \tau_0, \tau_1)$ be the output of $\mathcal{A}$ in the experiment. If $\mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}_0, I, \tau_0) = 1$, it holds that $\mathsf{tagpk} = g^{\mathsf{tagsk}_0}$ and $\tau_0 = H(I)^{\mathsf{tagsk}_0}$. Similarly, if $\mathsf{LIT.ChkTag}(\mathsf{tagpk}, \mathsf{tagsk}_1, I, \tau_1) = 1$, it holds that $\mathsf{tagpk} = g^{\mathsf{tagsk}_1}$ and

$\tau_1 = H(I)^{\mathsf{tagsk}_1}$. Since g is a generator of $\mathbb{G}_1$ and thus the order of g is p, the mapping $\mathbb{Z}_p \ni x \mapsto g^x \in \mathbb{G}_1$ is bijection. Then, the fact that $\mathsf{tagpk} = g^{\mathsf{tagsk}_0} = g^{\mathsf{tagsk}_1}$ implies that $\mathsf{tagsk}_0 = \mathsf{tagsk}_1$. Hence, if the above conditions hold, it holds that $\tau_0 = H(I)^{\mathsf{tagsk}_0} = H(I)^{\mathsf{tagsk}_1} = \tau_1$. For such a pair of inputs (τ_0, τ_1), $\mathsf{LIT.Link}(\tau_0, \tau_1)$ outputs 1. This concludes the proof.

Proof of Theorem E.5

Proof. Let $\mathcal{A}$ be a PPT adversary against the key secrecy of the scheme. Let us fix a security parameter n. We construct a reduction $\mathcal{B}$ that satisfies that $\mathsf{Adv}^{\mathsf{key\text{-}sec}}_{\mathsf{LIT},\mathcal{A}}(n) = \mathsf{Adv}^{\mathsf{DL}}_{\mathcal{G},\mathcal{B}}(n)$. The construction of $\mathcal{B}$ is as follows. Given a description of bilinear groups $(p, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T, e, g, h)$ and a DL problem y, $\mathcal{B}$ sets $\mathsf{tagpk} \leftarrow y$ and runs $\mathcal{A}(\mathsf{tagpk})$. We assume that any tag generation oracle query should be queried to the random oracle beforehand. When $\mathcal{A}$ issues a random oracle query I, $\mathcal{B}$ chooses $r_I \leftarrow \mathbb{Z}_p$ and returns g^{r_I}. When $\mathcal{A}$ issues a tag generation oracle query I, $\mathcal{B}$ retrieves r_I and returns tagpk^{r_I}. Once $\mathcal{A}$ outputs tagsk^* and terminates, $\mathcal{B}$ outputs tagsk^*.

This completes the description of $\mathcal{B}$. It holds that $\mathcal{B}$ perfectly simulate the experiment. Furthermore, if $\mathcal{A}$ wins the game, it holds that $y = \mathsf{tagpk} = g^{\mathsf{tagsk}^*}$. This means that if $\mathcal{A}$ wins the game, $\mathcal{B}$ solves the DL problem. This concludes the proof.

Proof of Theorem E.6

Proof. Let $\mathcal{A}$ be a PPT adversary against the key robustness of the scheme. Let $(I, \mathsf{tagpk}_0, \mathsf{tagsk}_0, \tau_0, \mathsf{tagpk}_1, \mathsf{tagsk}_1, \tau_1)$ be the output of $\mathcal{A}$. If $\mathsf{LIT.ChkTag}(\mathsf{tagpk}_0, \mathsf{tagsk}_0, I, \tau_0) = 1$, we have that $\mathsf{tagpk}_0 = g^{\mathsf{tagsk}_0}$ and $\tau_0 = H(I)^{\mathsf{tagsk}_0}$. Similarly, if $\mathsf{LIT.ChkTag}(\mathsf{tagpk}_1, \mathsf{tagsk}_1, I, \tau_1) = 1$, we have that $\mathsf{tagpk}_1 = g^{\mathsf{tagsk}_1}$ and $\tau_1 = H(I)^{\mathsf{tagsk}_1}$. If $\mathsf{LIT.Link}(\tau_0, \tau_1) = 1$, it holds that $H(I)^{\mathsf{tagsk}_0} = \tau_0 = \tau_1 = H(I)^{\mathsf{tagsk}_0}$. If these three conditions holds, with overwhelming probability, it holds that $\mathsf{tagsk}_0 = \mathsf{tagsk}_1$. This is because $H(I) \neq 1$, and hence $H(I)$ is a generator of $\mathbb{G}_1$. In that case, the mapping $\mathbb{Z}_p \ni x \mapsto H(I)^x \in \mathbb{G}_1$ is bijection and thus $\mathsf{tagsk}_0 = \mathsf{tagsk}_1$. The probability that $\mathcal{A}$ finds I satisfying $H(I) = 1$ is at most q_{h}/p where q_{h} is the number of random oracle queries from $\mathcal{A}$. This concludes the proof.

Accumulators with revocation. We present a construction of accumulator with revocation from the q-DHE assumption. They are abstracted from the Libert-Peters-Yung group signature scheme [28]. In this construction, we use the Libert-Yung construction of VC, which is presented in the appendix E. Next, we provide the instantiation of ACC using VC with the above instantiation. The following instantiation makes use of the subset difference method by Naor et al. [32] (See Section A.6).

$\mathsf{ACC.Setup}(1^n, m)$**.** Compute a commitment key $\mathsf{ck} \leftarrow \mathsf{VC.Setup}(1^n, m+1)$ of the vector commitment scheme and generate a key pair $(\mathsf{vk}_{\mathrm{cert}}, \mathsf{sk}_{\mathrm{cert}}) \leftarrow \mathsf{SIG.KG}(1^n)$. Then set $\mathsf{pp}_{\mathsf{acc}} \leftarrow (m, \mathsf{ck}, \mathsf{vk}_{\mathrm{cert}})$ and $\mathsf{sk}_{\mathsf{acc}} \leftarrow \mathsf{sk}_{\mathrm{cert}}$. Output $(\mathsf{pp}_{\mathsf{acc}}, \mathsf{sk}_{\mathsf{acc}})$.

$\mathsf{ACC.Wit}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{sk}_{\mathsf{acc}}, \mathsf{id})$**.** Compute $(\mathsf{com}, \mathsf{aux}) \leftarrow \mathsf{VC.Com}(\mathsf{ck}, (\mathsf{id}|_0, \mathsf{id}|_1, \ldots, \mathsf{id}|_m))$ and generate $\zeta \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{\mathrm{cert}}, \langle \mathsf{id}, \mathsf{com} \rangle)$. Set $\mathsf{wit} \leftarrow (\mathsf{com}, \mathsf{aux}, \zeta)$ and output wit.

$\mathsf{ACC.Acc}(\mathsf{pp}_{\mathsf{acc}}, R)$**.** Parse $(m, \mathsf{ck}, \mathsf{vk}_{\mathrm{cert}}) \leftarrow \mathsf{pp}_{\mathsf{acc}}$. Compute $((p_1, s_1), \ldots, (p_r, s_r)) \leftarrow \mathsf{SD}(1^m, R)$. Generate a key pair $(\mathsf{vk}_{\mathrm{revoke}}, \mathsf{sk}_{\mathrm{revoke}}) \leftarrow \mathsf{SIG.KG}(1^n)$ and generate a signature $\rho_i \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{\mathrm{revoke}}, \langle p_i, |p_i|, s_i, |s_i| \rangle)$ for all $i \in [r]$. Set $\mathsf{acc} \leftarrow \mathsf{vk}_{\mathrm{revoke}}$ and $\mathsf{aux}_{\mathsf{acc}} \leftarrow (\mathsf{vk}_{\mathrm{revoke}}, (p_i, s_i, \rho_i)_{i \in [r]})$ and output $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}})$.

$\mathsf{ACC.Prove}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{aux}_{\mathsf{acc}}, \mathsf{id}, \mathsf{wit})$**.** Parse $(m, \mathsf{ck}, \mathsf{vk}_{\mathrm{cert}}) \leftarrow \mathsf{pp}_{\mathsf{acc}}$, $(\mathsf{vk}_{\mathrm{revoke}}, (p_i, s_i, \rho_i)_{i \in [r]}) \leftarrow \mathsf{aux}_{\mathsf{acc}}$, and $(\mathsf{com}, \mathsf{aux}, \zeta) \leftarrow \mathsf{wit}$. Find $i \in [r]$ satisfying that p_i is a prefix of id and s_i is not a prefix of id. Compute openings $\mathsf{open}_1 \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \mathsf{aux}, |p_i| + 1)$ and $\mathsf{open}_2 \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \mathsf{aux}, |s_i| + 1)$. Set $\pi \leftarrow (\mathsf{com}, \zeta, p_i, |p_i|, s_i, |s_i|, \rho_i, \mathsf{id}|_{|s_i|}, \mathsf{open}_1, \mathsf{open}_2)$ and output wit.

$\mathsf{ACC.Verify}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{acc}, \mathsf{id}, \pi)$**.** Parse $(m, \mathsf{ck}, \mathsf{vk}_{\mathrm{cert}}) \leftarrow \mathsf{pp}_{\mathsf{acc}}$, $\mathsf{vk}_{\mathrm{revoke}} \leftarrow \mathsf{acc}$, and $(\mathsf{com}, \zeta, p, \ell_1, s, \ell_2, \rho, \hat{s}, \mathsf{open}_1, \mathsf{open}_2) \leftarrow \pi$. Check that $\mathsf{SIG.Ver}(\mathsf{vk}_{\mathrm{cert}}, \langle \mathsf{id}, \mathsf{com} \rangle, \zeta) = 1$, $\mathsf{SIG.Ver}(\mathsf{vk}_{\mathrm{revoke}}, \langle p, \ell_1, s, \ell_2 \rangle, \rho) = 1$, $\mathsf{VC.Open}(\mathsf{ck}, \mathsf{com}, \ell_1 + 1, p, \mathsf{open}_1) = 1$, $\mathsf{VC.Open}(\mathsf{ck}, \mathsf{com}, \ell_2 + 1, \hat{s}, \mathsf{open}_2) = 1$, and $s \neq \hat{s}$. If all the checks pass, output 1. Otherwise, output 0.

In the following, we provide the theorems with respect to the soundness. Proofs of the theorems are provided in the appendix E.

Theorem E.7. *If the signature scheme* SIG *is EUF-CMA secure and the vector commitment scheme* VC *is position binding, the pairing-based accumulator with revocation scheme is sound.*

Proof of Theorem E.7

Proof. Let $\mathcal{A}$ be a PPT adversary against the soundness of the pairing-based accumulator with revocation scheme. Fix a bound m of the number of users and a security parameter $n \in \mathbb{N}$. Let W be the event that $\mathcal{A}$ wins the game.

In the experiment of soundness, parsing the output of $\mathcal{A}$ as $(\mathsf{com}^*, \zeta^*, p^*, \ell_1^*, s^*, \ell_2^*, \rho^*, \hat{s}^*, \mathsf{open}_1^*, \mathsf{open}_2^*)$, we define the following events.

E_{cert}**.** The oracle $\mathcal{O}_{\mathrm{wit}}$ did not sign on $(\mathsf{id}^*, \mathsf{com}^*)$ at any point.

E_{revoke}**.** The experiment did not sign on $\langle p^*, \ell_1^*, s^*, \ell_2^* \rangle$ when generating a challenge.

E_{bind}**.** The event E_{cert} did not occur, and p^* is not a length-ℓ_1^* prefix of id^* or $\hat{s}^*$ is not a length-ℓ^* prefix of id^*.

Then we have the following inequality:

$$\Pr[W] \leq \Pr[W \wedge E_{\mathrm{cert}}] + \Pr[W \wedge E_{\mathrm{revoke}}] + \Pr[W \wedge E_{\mathrm{bind}}] + \Pr[W \wedge \neg E_{\mathrm{cert}} \wedge \neg E_{\mathrm{revoke}} \wedge \neg E_{\mathrm{bind}}].$$

We will bound each term one by one.

Lemma E.3. *There is an EUF-CMA adversary* $\mathcal{B}_1$ *satisfying* $\Pr[W \wedge E_{\mathrm{cert}}] \leq \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG}, \mathcal{B}_1}(n)$.

Proof. We construct $\mathcal{B}_1$ by internally running $\mathcal{A}$ and simulating the soundness experiment. The construction of $\mathcal{B}_1$ is as follows. Given a verification key vk of SIG, $\mathcal{B}_1$ computes $\mathsf{pp}_{\mathsf{acc}}$ following the experiment with an exception that $\mathsf{vk}_{\mathrm{cert}} \leftarrow \mathsf{vk}$ and runs $\mathcal{A}(\mathsf{pp}_{\mathsf{acc}})$. When $\mathcal{A}$ issues an $\mathcal{O}_{\mathrm{wit}}$ query id, $\mathcal{B}_1$ computes $(\mathsf{com}, \mathsf{aux}) \leftarrow \mathsf{VC.Com}(\mathsf{ck}, (\mathsf{id}|_0, \mathsf{id}|_1, \ldots, \mathsf{id}|_m))$ and sends $\langle \mathsf{id}, \mathsf{com} \rangle$ to the signing oracle. Receiving a signature ζ on $\langle \mathsf{id}, \mathsf{com} \rangle$, $\mathcal{B}_1$ sets $\mathsf{wit} \leftarrow (\mathsf{com}, \mathsf{aux}, \zeta)$ and returns wit to $\mathcal{A}$. When $\mathcal{A}$ issues a challenge oracle R, $\mathcal{B}_1$ responds to this following the experiment. When $\mathcal{A}$ outputs (id^*, π^*), $\mathcal{B}_1$ parses $(\mathsf{com}^*, \zeta^*, p^*, \ell_1^*, s^*, \ell_2^*, \rho^*, \hat{s}^*, \mathsf{open}_1^*, \mathsf{open}_2^*) \leftarrow \pi^*$ and outputs $(\langle \mathsf{id}^*, \mathsf{com}^* \rangle, \zeta^*)$.

This completes the description of $\mathcal{B}_1$. Then we bound the probability $\Pr[W \wedge E_{\mathrm{cert}}]$. Firstly, $\mathcal{B}_1$ perfectly simulates the experiment. If the event W occurs, the output of $\mathcal{B}_1$ is verified as valid. Furthermore, if the event E_{cert} occurs, $\mathcal{B}_1$ has not queried $\langle \mathsf{id}^*, \mathsf{com}^* \rangle$ to the signing oracle. Therefore, if $W \wedge E_{\mathrm{cert}}$ occurs, $\mathcal{B}_1$ wins the EUF-CMAgame. This implies the lemma.

Lemma E.4. *There is an EUF-CMA adversary* $\mathcal{B}_1$ *satisfying* $\Pr[W \wedge E_{\mathrm{cert}}] \leq \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SIG},\mathcal{B}_1}(n)$.

Proof. We construct $\mathcal{B}_2$ by internally running $\mathcal{A}$ and simulating the soundness experiment. The construction of $\mathcal{B}_2$ is as follows. Given a verification key vk, $\mathcal{B}_2$ sets up $\mathsf{pp}_{\mathsf{acc}}$ following the experiment and runs $\mathcal{A}(\mathsf{pp}_{\mathsf{acc}})$. When $\mathcal{A}$ issues an $\mathcal{O}_{\mathrm{wit}}$ query, $\mathcal{B}_2$ responds to this following the experiment. Since $\mathcal{B}_2$ set up $\mathsf{pp}_{\mathsf{acc}}$ by itself, $\mathcal{B}_2$ knows $\mathsf{sk}_{\mathsf{acc}}$ and thus $\mathcal{B}_2$ can compute ζ. When $\mathcal{A}$ issues a challenge query R, B_2 computes $((p_1, s_1), \ldots, (p_r, s_r)) \leftarrow \mathsf{SD}(1^m, R)$ and issues signing queries $\langle p_1, |p_1|, s_1, |s_1| \rangle$, $\ldots$, $\langle p_r, |p_r|, s_r, |s_r| \rangle$ and receives ρ_1, $\ldots$, ρ_r. Then $\mathcal{B}_2$ sets $\mathsf{acc} \leftarrow \mathsf{vk}$ and $\mathsf{aux}_{\mathsf{acc}} \leftarrow (\mathsf{vk}, ((p_i, s_i, \rho_i))_{i \in [r]})$ and returns $(\mathsf{acc}, \mathsf{aux}_{\mathsf{acc}})$. Once $\mathcal{A}$ outputs (id^*, π^*), $\mathcal{B}_2$ parses $(\mathsf{com}^*, \zeta^*, p^*, \ell_1^*, s^*, \ell_2^*, \rho^*, \hat{s}^*, \mathsf{open}_1^*, \mathsf{open}_2^*) \leftarrow \pi^*$ and outputs $(\langle p^*, \ell_1^*, s^*, \ell_2^* \rangle, \rho^*)$.

This completes the description of $\mathcal{B}_2$. Then we bound the probability $\Pr[W \wedge E_{\mathrm{revoke}}]$. Firstly, $\mathcal{B}_2$ perfectly simulates the experiment. If the event W occurs, the output of $\mathcal{B}_2$ is verified as valid under the verification key vk. Furthermore, if the event E_{revoke} occurs, $\mathcal{B}_2$ has not queried $\langle p^*, \ell_1^*, s^*, \ell_2^* \rangle$ to the signing oracle. Therefore, if $W \wedge E_{\mathrm{revoke}}$ occurs, $\mathcal{B}_2$ wins EUF-CMAgame. This implies the lemma.

Lemma E.5. *There is a position binding adversary* $\mathcal{B}_3$ *satisfying* $\Pr[W \wedge E_{\mathrm{bind}}] \leq \mathsf{Adv}^{\mathsf{pos\text{-}bind}}_{\mathsf{VC},\mathcal{B}_3}(n)$.

Proof. We construct $\mathcal{B}_3$ by internally running $\mathcal{A}$ and simulating the soundness experiment. The construction of $\mathcal{B}_3$ is as follows. Given a commitment key ck, $\mathcal{B}_3$ sets up $\mathsf{pp}_{\mathsf{acc}}$ by using this given ck as a part of $\mathsf{pp}_{\mathsf{acc}}$. Then $\mathcal{B}_3$ runs $\mathcal{A}(\mathsf{pp}_{\mathsf{acc}})$ by responding to queries from $\mathcal{A}$ honestly. When responding to $\mathcal{O}_{\mathrm{wit}}$ queries, $\mathcal{B}_3$ stores $(\mathsf{com}, \mathsf{aux})$ computed in response to queries for later purpose. Once $\mathcal{A}$ outputs (id^*, π^*) and terminates, $\mathcal{B}_3$ parses $(\mathsf{com}^*, \zeta^*, p^*, \ell_1^*, s^*, \ell_2^*, \rho^*, \hat{s}^*, \mathsf{open}_1^*, \mathsf{open}_2^*) \leftarrow \pi^*$. Then $\mathcal{B}_3$ searches for an $\mathcal{O}_{\mathrm{wit}}$ query id satisfying that $\mathsf{id} = \mathsf{id}^*$ and the com computed in response to this query is equal to com^* and retrieves aux that is generated together with com. Then $\mathcal{B}_3$ checks whether p^* is not the length-ℓ_1^*

prefix of id^*. If not, $\mathcal{B}_3$ computes $\mathsf{open}_1 \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \mathsf{aux}, \ell_1^* + 1)$ and outputs $(\mathsf{com}^*, \ell_1^* + 1, \mathsf{id}^*|_{\ell_1^*}, \mathsf{open}_1, p^*, \mathsf{open}_1^*)$. If it is, $\mathcal{B}_3$ checks whether $\hat{s}^*$ is not the length-ℓ_2^* prefix of id^*. If not, $\mathcal{B}_3$ computes $\mathsf{open}_2 \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \mathsf{aux}, \ell_2^* + 1)$ and outputs $(\mathsf{com}^*, \ell_2^* + 1, \mathsf{id}^*|_{\ell_1^*}, \mathsf{open}_2, \hat{s}^*, \mathsf{open}_2^*)$. If it is, outputs $(\perp, \perp, \perp, \perp, \perp, \perp)$.

This completes the description of $\mathcal{B}_3$. Then we bound the probability $\Pr[W \wedge E_{\text{bind}}]$. Let us assume $W \wedge E_{\text{bind}}$ occurs. Since E_{bind} occurs, we have $\neg E_{\text{cert}}$ occur. Then $\mathcal{B}_3$ can find an $\mathcal{O}_{\text{wit}}$ query id which is equal to id^* and com that is generated in response to this query is equal to com^*. Since E_{bind} occurs, $\mathcal{B}_3$ finds one of the following two conditions holds: p^* is not the length-ℓ_1^* prefix of id^*, or s^* is not the length-ℓ_2^* prefix of id^*. If the former is true, $\mathcal{B}_3$ outputs $(\mathsf{com}^*, \ell_1^* + 1, \mathsf{id}^*|_{\ell_1^*}, \mathsf{open}_1, p^*, \mathsf{open}_1^*)$. Due to the correctness of VC, it holds that $\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, \ell_1^* + 1, \mathsf{id}|_{\ell_1^*}, \mathsf{open}_1) = \mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}^*, \ell_1^* + 1, \mathsf{id}^*|_{\ell_1^*}, \mathsf{open}_1) = 1$. Due to the event W, it holds that $\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}^*, \ell_1^* + 1, p^*, \mathsf{open}_1^*) = 1$. Thus, in this case, $\mathcal{B}_3$ breaks the position bindingproperty of VC. In the latter case, $\mathcal{B}_3$ outputs $(\mathsf{com}^*, \ell_2^* + 1, \mathsf{id}^*|_{\ell_1^*}, \mathsf{open}_2, \hat{s}^*, \mathsf{open}_2^*)$. Similarly, we have that $\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, \ell_2^* + 1, \mathsf{id}|_{\ell_2^*}, \mathsf{open}_2) = \mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}^*, \ell_2^* + 1, \mathsf{id}^*|_{\ell_2^*}, \mathsf{open}_2) = 1$ and $\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}^*, \ell_2^* + 1, \hat{s}^*, \mathsf{open}_2^*) = 1$. In either case, $\mathcal{B}_3$ breaks the position bindingproperty, which implies the lemma.

Lemma E.6. *It holds that* $\Pr[W \wedge \neg E_{\text{cert}} \wedge \neg E_{\text{revoke}} \wedge \neg E_{\text{bind}}] = 0$.

Proof. Let us assume that the event $W \wedge \neg E_{\text{cert}} \wedge \neg E_{\text{revoke}} \wedge \neg E_{\text{bind}}$ occurs. Then, due to the event $\neg E_{\text{revoke}}$, we have that the subset described by (p^*, s^*) does not include id^*. Besides, since $\neg E_{\text{bind}}$ occurs, p^* is the length-ℓ_1^* prefix of id^* and $\hat{s}^*$ is the length-ℓ_2^* prefix of id^*. Furthermore, s^* is length ℓ_2^* and $\hat{s}^* \neq s^*$, which implies that id^* does not have s^* as a prefix. This means that id^* is included in the subset described by (p^*, s^*). This contradicts to the event E_{revoke} and hence the probability in the lemma is 0.

These lemmas prove the inequality and hence conclude the proof.

Instantiation of VC from the q-DHE assumption. Here, we provide a construction of q-DHE assumption by Libert and Yung [29].

$\mathsf{VC.Setup}(1^n, q)$**.** Compute $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^n)$. Choose $z \leftarrow \mathbb{Z}_p$ and sets $g_i \leftarrow g^{z^i}$ and $h_i \leftarrow h^{z^i}$ for $i \in [2q]\backslash\{q+1\}$. Output $\mathsf{ck} \leftarrow ((g_i)_{i\in[2q]\backslash\{q+1\}}, (h_i)_{i\in[2q]\backslash\{q+1\}})$.

$\mathsf{VC.Com}(\mathsf{ck}, (m_1, \ldots, m_q))$**.** Compute $\mathsf{com} \leftarrow \prod_{i\in[q]} g_{q+1-i}^{m_i}$ and $\mathsf{aux} \leftarrow (m_1, \ldots, m_q)$ and output $(\mathsf{com}, \mathsf{aux})$.

$\mathsf{VC.Open}(\mathsf{ck}, \mathsf{aux}, j)$**.** Compute $\mathsf{open} \leftarrow \prod_{i\in[q]\backslash\{j\}} g_{q+1-i+j}^{m_j}$ and output open.

$\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, i, m, \mathsf{open})$**.** Verify $e(\mathsf{com}, g_i) = e(\mathsf{open}, h)e(g_1, h_q)^m$. If it holds, output 1. Otherwise, output 0.

Theorem E.8. *If the symmetric q-Diffie-Hellman exponent assumption for $\mathcal{G}$ holds, the above vector commitment scheme is position binding.*

Non-Interactive Zero-Knowledge for $\boldsymbol{\rho_1}$. We need a non-interactive zero-knowledge for the following language:

$$\begin{aligned}\rho_1 = \{(((c_1,c_2), (\mathsf{vk}, \mathsf{pp}_{\mathsf{acc}}), (\mathsf{ek}_1, \mathsf{ek}_2), \mathsf{acc}, \mathsf{item}, \tau), (\mathsf{upk}, \mathsf{usk}, \mathsf{id}, \pi_{\mathsf{acc}}, \theta, r_1, r_2)) \mid \\ c_1 = \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1) \wedge c_2 = \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2) \\ \wedge \mathsf{LIT.ChkKey}(\mathsf{upk}, \mathsf{usk}) = 1 \\ \wedge \mathsf{LIT.ChkTag}(\mathsf{upk}, \mathsf{usk}, \langle \mathsf{vk}, \mathsf{pp}_{\mathsf{acc}}, \mathsf{item}\rangle, \tau) = 1 \\ \wedge \mathsf{ACC.Verify}(\mathsf{pp}_{\mathsf{acc}}, \mathsf{acc}, \mathsf{id}, \pi_{\mathsf{acc}}) = 1 \\ \wedge \mathsf{SIG.Ver}(\mathsf{vk}, \langle \mathsf{upk}, \mathsf{id}, \mathsf{item}\rangle, \theta) = 1\}.\end{aligned}$$

This condition is equivalent to the following:

$$\begin{aligned}&c_1 = \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{upk}; r_1),\\ &c_2 = \mathsf{PKE.Enc}(\mathsf{ek}_2, \mathsf{upk}; r_2),\\ &\mathsf{LIT.ChkKey}(\mathsf{upk}, \mathsf{usk}) = 1,\\ &\mathsf{LIT.ChkTag}(\mathsf{upk}, \mathsf{usk}, \langle \mathsf{ipk}, \mathsf{item}\rangle, \tau) = 1,\\ &\mathsf{SIG.Ver}(\mathsf{vk}_{\mathrm{cert}}, \langle \mathsf{id}, \mathsf{com}\rangle, \zeta) = 1,\\ &\mathsf{SIG.Ver}'(\mathsf{vk}_{\mathrm{revoke}}, \langle p, \ell_1, s, \ell_2, \rangle, \rho) = 1,\\ &\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, \ell_1 + 1, p, \mathsf{open}_1) = 1,\\ &\mathsf{VC.Ver}(\mathsf{ck}, \mathsf{com}, \ell_2 + 1, \hat{s}, \mathsf{open}_2) = 1,\\ &s \neq \hat{s},\\ &\mathsf{SIG.Ver}(\mathsf{vk}, \langle \mathsf{upk}, \mathsf{id}, \mathsf{item}\rangle, \theta) = 1\end{aligned}$$

where $\mathsf{pp}_{\mathsf{acc}} = (n, \mathsf{ck}, \mathsf{vk}_{\mathrm{cert}})$, $\mathsf{acc} = \mathsf{vk}_{\mathrm{revoke}}$, and $\pi_{\mathsf{acc}} = (\mathsf{com}, \zeta, p, \ell_1, s, \ell_2, \rho, \hat{s}, \mathsf{open}_1, \mathsf{open}_2)$. By substituting the equations with the instantiation, we can rewrite the equation as follows:

$$\begin{aligned}&e_1 = g^{r_1},\\ &e_2 = g^{r_2},\\ &f_1/y_1^{r_1} = f_2/y_2^{r_2},\\ &f_1/y_1^{r_1} = g^{\mathsf{usk}},\\ &\tau = H(\langle \mathsf{ipk}, \mathsf{item}\rangle)^{\mathsf{usk}},\\ &e(V, R) = e(g, S),\\ &e(T, R)e(h^{\mathsf{id}}, U_1)e(\mathsf{com}, U_2) = e(g, h),\\ &e(R', V') = e(S', h),\\ &e(R', T')e(U_1', h^p)e(U_2', h_{\ell_1+1})e(U_3', h^s)e(U_4', h_{\ell_2} + 1) = e(g, h),\\ &e(\mathsf{com}, h_{\ell_1+1}) = e(\mathsf{open}_1, h)e(g_1, h_{\ell+1})^p,\\ &e(\mathsf{com}, h_{\ell_2+1}) = e(\mathsf{open}_2, h)e(g_1, h_{\ell+1})^{\hat{s}},\\ &s \neq \hat{s},\\ &e(V'', R'') = e(g, S''),\end{aligned}$$

$$e(T'', R'')e(f_1/y_1^{r_1}, U_1'')e(g^{\mathsf{id}}, U_2'')e(g^{\mathsf{item}}, U_3'') = e(g, h)$$

where $c_1 = (e_1, f_1)$, $c_2 = (e_2, f_2)$, $\mathsf{ek}_1 = y_1$, $\mathsf{ek}_2 = y_2$, $\mathsf{vk}_{\mathrm{cert}} = (U_1, U_2, V)$, $\zeta = (R, S, T)$, $\mathsf{vk}_{\mathrm{revoke}} = (U_1', \dots, U_4', V')$, $\rho = (R', S', T')$, $\mathsf{ck} = ((g_i)_{i\in[2(\ell+1)]\setminus\{\ell+2\}}, (h_i)_{i\in[2(\ell+1)]\setminus\{\ell+2\}})$, $\mathsf{vk} = (U_1'', \dots, U_3'', V'')$, and $\theta = (R'', S'', T'')$. Here, the equation $\mathsf{LIT.ChkKey}(\mathsf{upk}, \mathsf{usk}) = 1$, which is instantiated as $\mathsf{upk} = g^{\mathsf{usk}}$, is replaced with $f_1/y_1^{u_1} = g^{\mathsf{usk}}$ for the consistency between this equation and the ElGamal encryption. Similarly, the appearance of upk in the verification equations of θ is also replaced with $f_1/y_1^{r_1}$. In addition, note that in order to sign on $\ell_1 + 1$ and $\ell_2 + 1$ by ρ, we instead sign on h_{ℓ_1+1} and h_{ℓ_2+1}.

We explain how these equations are proved by a Schnorr-like protocol. Linear equations are easy to adapt to a Schnorr-like protocol (e.g., by Maurer's abstraction [30]) but non-linear equations are not.

We explain how these non-linear equations are adapted to a Schnorr-like protocol.

To this end, we let the protocol send the following encryption of some witnesses and the random element:

$$\begin{array}{llll} C_0 \leftarrow g^{\rho}, & C_1 \leftarrow \mathsf{pk}_1^{\rho}\mathsf{com} & C_2 \leftarrow \mathsf{pk}_2^{\rho} g^{s}, & C_3 \leftarrow \mathsf{pk}_3^{\rho} g^{\hat{s}}, \\ D_0 \leftarrow h^{\sigma}, & D_1 \leftarrow (\mathsf{pk}_1')^{\sigma} h_{\ell_1+1}, & D_2 \leftarrow (\mathsf{pk}_2')^{\sigma} h_{\ell_2+1}, & E \leftarrow g^{\upsilon(s-\hat{s})}. \end{array}$$

where ρ, $\sigma \leftarrow \mathbb{Z}_p$, $\upsilon \leftarrow \mathbb{Z}_p^*$, and the public keys pk_1, pk_2, $\mathsf{pk}_3 \in \mathbb{G}_1$ and pk_1', $\mathsf{pk}_2' \in \mathbb{G}_2$ are chosen by a random oracle. Here, we need to prove the well-formedness of the part of the ciphertext C_0, C_2, C_3, and D_0:

$$C_0 = g^{\rho}, \qquad C_2 = \mathsf{pk}_2^{\rho} g^{s}, \qquad C_3 = \mathsf{pk}_3^{\rho} g^{\hat{s}}, \qquad D_0 = h^{\sigma}.$$

Furthermore, we let the protocol rerandomize (R, S, T) and (R', S', T') by running $(\tilde{R}, \tilde{S}, \tilde{T}) \leftarrow \mathsf{SIG.Rerand}(\mathsf{vk}_{\mathrm{cert}}, (R, S, T))$, $(\tilde{R}', \tilde{S}', \tilde{T}') \leftarrow \mathsf{SIG.Rerand}'(\mathsf{vk}_{\mathrm{revoke}}, (R', S', T'))$, and $(\tilde{R}'', \tilde{S}'', \tilde{T}'') \leftarrow \mathsf{SIG.Rerand}(\mathsf{vk}, (R'', S'', T''))$ and prove

$$\begin{aligned} & e(V, \tilde{R}) = e(g, \tilde{S}), \\ & e(\tilde{T}, \tilde{R})e(h^{\mathsf{id}}, U_1)e(\mathsf{com}, U_2) = e(g, h), \\ & e(\tilde{R}', V') = e(\tilde{S}', h), \\ & e(\tilde{R}', \tilde{T}')e(U_1', h^{p})e(U_2', h_{\ell_1+1})e(U_3', h^{s})e(U_4', h_{\ell_2} + 1) = e(g, h), \\ & e(V'', \tilde{R}'') = e(g, \tilde{S}''), \\ & e(\tilde{T}'', \tilde{R}'')e(f_1/y_1^{r_1}, U_1'')e(g^{\mathsf{id}}, U_2'')e(g^{\mathsf{item}}, U_3'') = e(g, h) \end{aligned}$$

instead. This way, we can send $\tilde{R}$, $\tilde{R}'$, and $\tilde{R}''$ in the clear. In addition, for consistency between the above equation and the encryption C_1, D_1, and D_2, we need to replace the appearance of com, h_{ℓ_1+1}, and h_{ℓ_2+1} with the ciphertexts:

$$\begin{aligned} & e(V, \tilde{R}) = e(g, \tilde{S}), \\ & e(\tilde{T}, \tilde{R})e(h^{\mathsf{id}}, U_1)e(C_1/\mathsf{pk}_1^{\rho}, U_2) = e(g, h), \end{aligned}$$

$$e(\tilde{R}', V') = e(\tilde{S}', h),$$
$$e(\tilde{R}', \tilde{T}')e(U_1', h^p)e(U_2', D_1/(\mathsf{pk}_1')^\sigma)e(U_3', h^s)e(U_4', D_2/(\mathsf{pk}_2')^\sigma) = e(g, h)$$

These equations are now linear. Then we can adapt these to a Schnorr-type protocol.[15]

Then the remaining non-linear equations are as follows:

$$e(\mathsf{com}, h_{\ell_1+1}) = e(\mathsf{open}_1, h)e(g_1, h_{\ell+1})^p,$$
$$e(\mathsf{com}, h_{\ell_2+1}) = e(\mathsf{open}_2, h)e(g_1, h_{\ell+1})^{\hat{s}},$$
$$s \neq \hat{s}.$$

We explain how we prove the above three equations.

The first equation can be rewritten by substituting com and h_{ℓ_1+1} by their encryption as follows:

$$e(C_1/\mathsf{pk}_1^\rho, D_1/(\mathsf{pk}_1')^\sigma) = e(\mathsf{open}_1, h)e(g_1, h_{\ell+1})^p.$$

This equation is equivalent to

$$e(C_1, D_1)e(C_1, \mathsf{pk}_1')^\sigma e(\mathsf{pk}_1, D_1)^\rho e(\mathsf{pk}_1, \mathsf{pk}_1')^{\delta_{\rho\sigma}} = e(\mathsf{open}_1, h)e(g_1, h_{\ell+1})^p,$$

if we substitute $\rho\sigma = \delta_{\rho\sigma}$. We need to ensure that $\delta_{\rho\sigma} = \rho\sigma$, which can be done by proving the additional equation

$$C_0^\sigma = g^{\delta_{\rho\sigma}}.$$

Similarly, the second equation can be rewritten as

$$e(C_1, D_2)e(C_1, \mathsf{pk}_2')^\sigma e(\mathsf{pk}_1, D_2)^\rho e(\mathsf{pk}_1, \mathsf{pk}_2')^{\delta_{\rho\sigma}} = e(\mathsf{open}_2, h)e(g_1, h_{\ell+1})^{\hat{s}}.$$

Finally, the equation $s \neq \hat{s}$ is rewritten as follows. Note we included the element $E = g^{\upsilon(s-\hat{s})}$. Then if we can prove the well-formedness of E, the remaining task is to check that $E \neq 1$. Since E is sent in the clear, this check is easy. To prove the well-formedness of E, we introduce variables $\delta_{\upsilon(s-\hat{s})} = \upsilon(s-\hat{s})$ and $\delta_{\upsilon\rho} = \upsilon\rho$ to linearize the well-formedness of E. Then we prove the following equations:

$$E = g^{\delta_{\upsilon(s-\hat{s})}}, \qquad (C_2/C_3)^\upsilon = (\mathsf{pk}_2/\mathsf{pk}_3)^{\delta_{\upsilon\rho}} g^{\delta_{\upsilon(s-\hat{s})}}, \qquad C_0^\upsilon = g^{\delta_{\upsilon\rho}}.$$

To summarize, our Schnorr-like protocol proceeds as follows. Firstly, the prover sends C_0, C_1, C_2, C_3, D_0, D_1, D_2, and E computed as above. Then the

[15] We use some group elements as a part of the witnesses of a Shcnorr-like protocol. This is not a very popular approach but it completely fits into Maurer's abstraction. For example, if we prove the knowledge of $\tilde{S}$ satisfying $e(V, \tilde{R}) = e(g, \tilde{S})$ with public V, $\tilde{R}$, and g, we simply consider a one-way group homomorphism $f\colon \tilde{S} \mapsto e(g, \tilde{S})$ and prove the knowledge of the preimage $\tilde{S}$ such that $e(V, \tilde{R}) = f(\tilde{S})$ following Maurer's abstraction.

prover and the verifier conduct a Schnorr-like protocol for proving the following linear equations:

$$
\begin{aligned}
&C_0 = g^{\rho}, \\
&C_2 = \mathsf{pk}_2^{\rho} g^{s}, \\
&C_3 = \mathsf{pk}_3^{\rho} g^{\hat{s}}, \\
&D_0 = h^{\sigma}, \\
&e_1 = g^{r_1}, \\
&e_2 = g^{r_2}, \\
&f_1 / y_1^{r_1} = f_2 / y_2^{r_2}, \\
&f_1 / y_1^{r_1} = g^{\mathsf{usk}}, \\
&\tau = H(\langle \mathsf{ipk}, \mathsf{item} \rangle)^{\mathsf{usk}}, \\
&e(V, \tilde{R}) = e(g, \tilde{S}), \\
&e(\tilde{T}, \tilde{R}) e(h^{\mathsf{id}}, U_1) e(C_1 / \mathsf{pk}_1^{\rho}, U_2) = e(g, h), \\
&e(\tilde{R}', V') = e(\tilde{S}', h), \\
&e(\tilde{R}', \tilde{T}') e(U_1', h^{p}) e(U_2', D_1 / (\mathsf{pk}_1')^{\sigma}) e(U_3', h^{s}) e(U_4', D_2 / (\mathsf{pk}_2')^{\sigma}) = e(g, h), \\
&e(C_1, D_1) e(C_1, \mathsf{pk}_1')^{\sigma} e(\mathsf{pk}_1, D_1)^{\rho} e(\mathsf{pk}_1, \mathsf{pk}_1')^{\delta_{\rho\sigma}} = e(\mathsf{open}_1, h) e(g_1, h_{\ell+1})^{p}, \\
&e(C_1, D_2) e(C_1, \mathsf{pk}_2')^{\sigma} e(\mathsf{pk}_1, D_2)^{\rho} e(\mathsf{pk}_1, \mathsf{pk}_2')^{\delta_{\rho\sigma}} = e(\mathsf{open}_2, h) e(g_1, h_{\ell+1})^{\hat{s}}, \\
&e(V'', \tilde{R}'') = e(g, \tilde{S}''), \\
&e(\tilde{T}'', \tilde{R}'') e(f_1 / y_1^{r_1}, U_1'') e(g^{\mathsf{id}}, U_2'') e(g^{\mathsf{item}}, U_3'') = e(g, h), \\
&C_0^{\sigma} = g^{\delta_{\rho\sigma}}, \\
&E = g^{\delta_{\upsilon(s-\hat{s})}}, \\
&(C_2 / C_3)^{\upsilon} = (\mathsf{pk}_2 / \mathsf{pk}_3)^{\delta_{\upsilon\rho}} g^{\delta_{\upsilon(s-\hat{s})}}, \\
&C_0^{\upsilon} = g^{\delta_{\upsilon\rho}}.
\end{aligned}
$$

If the protocol ends with acceptance and $E \neq 1$, the verifier accepts the statement. Otherwise, the verifier rejects it.

In addition, we need the simulation extractability of NIZK in our generic construction. It is not necessarily true that the Fiat-Shamir transformation of any Σ protocol is simulation extractable but there is a simple way to enhance the Fiat-Shamir transformation in such a way that any Σ protocol is transformed into a simulation extractable NIZK. The idea is that, when generating a non-interactive proof, a prover generates a key pair of a strongly unforgeable one-time signature scheme, appends the verification key to the input to the hash function for generating a challenge, and signs on the response. The resulting proof consists of the standard Fiat-Shamir transformed proof together with the verification key and a signature. A verifier verifies the Fiat-Shamir transformed proof *and* the one-time signature and accepts the entire proof if both of the verifications pass. Detailed description will be provided in the supplemental materials F.

Non-Interactive Zero-Knowledge for ρ_2. We then explain how we instantiated the NIZK for ρ_2. The statement that needs to be proven is as follows:

$$\rho_2 = \{((\mathsf{ek}_1, c_1, \mathsf{upk}), (\mathsf{dk}_1, r_{\mathsf{PKE}})) \mid \\ \mathsf{upk} = \mathsf{PKE.Dec}(\mathsf{dk}_1, c_1) \wedge (\mathsf{ek}_1, \mathsf{dk}_1) = \mathsf{PKE.KG}(1^n; r_{\mathsf{PKE}})\}.$$

This statement is instantiated with the above Cramer-Shoup encryption as follows:

$$f_1/e_1^{x_1} = \mathsf{upk}, \\ y_1 = g^{x_1}.$$

where $\mathsf{ek}_1 = y_1 = g^{x_1}$, $c_1 = (e_1, f_1)$, and $\mathsf{dk}_1 = x_1$.

These relations are all linear. Then we can instantiate the NIZK by simply applying Maurer's generic protocol [30].

Signature Size. A signature of our ARS scheme includes two ElGamal ciphertexts in $\mathbb{G}_1$, one linkable indistinguishable tag in $\mathbb{G}_1$, a Fiat-Shamir NIZK, and a verification key and a signature of a one-time signature scheme. The two ElGamal ciphertexts requires four $\mathbb{G}_1$ elements and the likable indistinguisable tag requires one $\mathbb{G}_1$ element. The language that our Fiat-Shamir NIZK proves has three $\mathbb{G}_1$ witnesses, three $\mathbb{G}_2$ elements, and 12 $\mathbb{Z}_p$ witnesse. The response of the underlying Σ protocol includes the same number of elements as above. In addition to this, one $\mathbb{Z}_p$ challenge is needed. Finally, for the one-time signature, we use Bellare-Shoup's transformation [8] applied to the Okamoto identification [36]. This one-time signature scheme has a verification key of three $\mathbb{G}_1$ elements and a signature of two $\mathbb{Z}_p$ element. In total, a signature of our scheme includes 16 $\mathbb{G}_1$ elements, 6 $\mathbb{G}_2$ elements, and 15 $\mathbb{Z}_p$ elements. This requires in total 16 kilobits for one signature if we use a BLS12-381 curve.

F Fiat-Shamir Transformation with Simulation Extractability

In our generic construction, we need a simulation extractable NIZK. Here, we explain how we generically obtain a simulation extractable NIZK by the Fiat-Shamir transformation.

We first define a syntax of Σ protocols with recoverable commitment, which will be then transformed into a simulation extractable NIZK. A Σ protocol with recoverable commitment for an NP relation $\mathcal{R}$ is defined by the following algorithms.

$\Sigma.\mathsf{Setup}(1^n) \to \mathsf{crs}$. The setup algorithm, given a security parameter 1^n, outputs a common reference string crs.

$\Sigma.\mathsf{Com}(\mathsf{crs}, \mathsf{X}, \mathsf{W}) \to (\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \mathsf{state})$. The commitment algorithm, given a common reference string crs, a statement X, and a W, outputs a commitment $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}})$ and state information state.

$\Sigma.\mathsf{Chal}(\mathsf{crs},\mathsf{X}) \to \beta$**.** The challenge algorithm, given a common reference string crs and a statement X, outputs a challenge β.

$\Sigma.\mathsf{Resp}(\mathsf{crs},\mathsf{X},\mathsf{W},\beta,\mathsf{state}) \to \gamma$**.** The response algorithm, given a common reference string crs, a statement X, a witness W, a challenge β, and state information state, outputs a response γ.

$\Sigma.\mathsf{Verify}(\mathsf{crs},\mathsf{X},\alpha_{\mathrm{unrec}},\beta,\gamma)$**.** The verification algorithm, given a common reference string crs, a statement X, the unrecoverable part of a commitment α_{unrec}, a challenge β, and a response γ, outputs a recovered commitment α_{rec} or a rejection symbol $\bot$.

We require a Σ protocol to be correct, honest-verifier zero-knowledge, specially sound, and high min-entropy commitments.

Definition F.1 (Correctness). *A Σ protocol Σ satisfies correctness if for all $n \in \mathbb{N}$ and all $(\mathsf{X},\mathsf{W}) \in \mathcal{R}$, we have*

$$\Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \Sigma.\mathsf{Setup}(1^n), \\ (\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}},\mathsf{state}) \leftarrow \Sigma.\mathsf{Com}(\mathsf{crs},\mathsf{X},\mathsf{W}), \\ \beta \leftarrow \Sigma.\mathsf{Chal}(\mathsf{crs},\mathsf{X}), \\ \gamma \leftarrow \Sigma.\mathsf{Resp}(\mathsf{crs},\mathsf{X},\mathsf{W},\beta,\mathsf{state}), \\ \bar{\alpha}_{\mathrm{rec}} \leftarrow \Sigma.\mathsf{Verify}(\mathsf{crs},\mathsf{X},\alpha_{\mathrm{unrec}},\beta,\gamma) \end{array} : \alpha_{\mathrm{rec}} = \bar{\alpha}_{\mathrm{rec}} \neq \bot \right] = 1.$$

Definition F.2 (Honest-verifier zero-knowledge). *A Σ protocol Σ satisfies honest-verifier zero-knowledge if there is simulator algorithms $\Sigma.\mathsf{Setup}$ and $\Sigma.\mathsf{Sim}$ satisfying that for all PPT adversary $\mathcal{A}$, it holds that*

$$\left| \Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \Sigma.\mathsf{Setup}(1^n), \\ b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{prf}}}(\mathsf{crs}) \\ : b = 1 \end{array}\right] - \Pr\left[\begin{array}{c} (\mathsf{crs},\mathsf{td}) \leftarrow \Sigma.\mathsf{SimSetup}(1^n), \\ b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{sim}}}(1^n), \\ : b = 1 \end{array}\right] \right| = \mathsf{negl}(n)$$

where the oracles are defined as follows.

$\mathcal{O}_{\mathrm{prf}}$**.** *Given a pair $(\mathsf{X},\mathsf{W}) \in \mathcal{R}$, this oracle computes $(\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}},\mathsf{state}) \leftarrow \Sigma.\mathsf{Com}(\mathsf{crs},\mathsf{X},\mathsf{W})$, $\beta \leftarrow \Sigma.\mathsf{Chal}(\mathsf{crs},\mathsf{X})$, and $\gamma \leftarrow \Sigma.\mathsf{Resp}(\mathsf{crs},\mathsf{X},\mathsf{W},\beta,\mathsf{state})$ and returns $(\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}},\beta,\gamma)$. If $(\mathsf{X},\mathsf{W}) \notin \mathcal{R}$, the oracle returns $\bot$.*

$\mathcal{O}_{\mathrm{sim}}$**.** *Given a pair $(\mathsf{X},\mathsf{W}) \in \mathcal{R}$, this oracle computes $(\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}},\beta,\gamma) \leftarrow \Sigma.\mathsf{Sim}(\mathsf{td},\mathsf{X})$ and returns $(\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}},\beta,\gamma)$. If $(\mathsf{X},\mathsf{W}) \notin \mathcal{R}$, the oracle returns $\bot$.*

Definition F.3 (Special soundness). *A Σ protocol Σ is specially sound, if there exists an extractor $\Sigma.\mathsf{Ext}$ satisfying that for any common reference string crs, any statement X, and any related transcripts $(\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}},\beta,\gamma)$ and $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}},\beta',\gamma')$, if it holds that $\Sigma.\mathsf{Verify}(\mathsf{crs},\alpha_{\mathrm{unrec}},\beta,\gamma) = \alpha_{\mathrm{rec}} \neq \bot$, $\Sigma.\mathsf{Verify}(\mathsf{crs}, \alpha_{\mathrm{unrec}},\beta',\gamma') = \alpha_{\mathrm{rec}} \neq \bot$, and $\beta \neq \beta'$, then the output $w \leftarrow \Sigma.\mathsf{Ext}(\mathsf{crs},\mathsf{X},\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}},\beta,\gamma,\beta',\gamma')$ satisfies that $(\mathsf{X},\mathsf{W}) \in \mathcal{R}$.*

Definition F.4 (High min-entropy of commitments). *Let Σ is a Σ protocol. We define $\mu(\mathsf{crs},\mathsf{X},\mathsf{W}) = \max_{\bar{\alpha}} \Pr[(\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}},-) \leftarrow \Sigma.\mathsf{Com}(\mathsf{crs},\mathsf{X},\mathsf{W}) : \bar{\alpha} = (\alpha_{\mathrm{unrec}},\alpha_{\mathrm{rec}})]$ and $\varepsilon(n) = \min_{(\mathsf{crs},\mathsf{X},\mathsf{W})}(-\log_2 \mu(\mathsf{crs},\mathsf{X},\mathsf{W}))$. The Σ protocol Σ has high min-entropy of commitments if $\varepsilon(n)$ is super-logarithmic.*

A Σ protocol with recoverable commitment is transformed to a simulation extractable NIZK in the random oracle model. We use a strongly unforgeable one-time signature scheme $(\mathsf{SIG.KG}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$.

$\mathsf{NIZK.Setup}(1^n)$. On input a security parameter 1^n, set $\mathsf{crs} \leftarrow \Sigma.\mathsf{Setup}(1^n)$ and output crs.

$\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$. Compute $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \mathsf{state}) \leftarrow \Sigma.\mathsf{Com}(\mathsf{crs}, \mathsf{X}, \mathsf{W})$ and generate a key pair $(\mathsf{vk}_{\mathrm{ot}}, \mathsf{sk}_{\mathrm{ot}}) \leftarrow \mathsf{SIG.KG}(1^n)$. Compute $\beta \leftarrow \mathcal{H}(\mathsf{crs}, \mathsf{X}, \mathsf{lbl}, \mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}})$, $\gamma \leftarrow \Sigma.\mathsf{Resp}(\mathsf{crs}, \mathsf{X}, \mathsf{W}, \beta, \mathsf{state})$, and $\sigma_{\mathrm{ot}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{\mathrm{ot}}, \gamma)$. Output $\pi \leftarrow (\mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \beta, \gamma, \sigma_{\mathrm{ot}})$.

$\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{lbl}, \mathsf{X}, \pi)$. Parse $(\mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \beta, \gamma, \sigma_{\mathrm{ot}}) \leftarrow \pi$. Compute $\bar{\alpha}_{\mathrm{rec}} \leftarrow \Sigma.\mathsf{Verify}(\mathsf{crs}, \mathsf{X}, \alpha_{\mathrm{unrec}}, \beta, \gamma)$ and check that $\bar{\alpha}_{\mathrm{rec}} \neq \bot$, $\beta = \mathcal{H}(\mathsf{crs}, \mathsf{X}, \mathsf{lbl}, \mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$, and $\mathsf{SIG.Ver}(\mathsf{vk}_{\mathrm{ot}}, \gamma, \sigma_{\mathrm{ot}}) = 1$. If all the checks pass, output 1. Otherwise, output 0.

We prove that the above NIZK is zero-knowledge and simulation extractable.

Theorem F.1 (Zero-knowledge). *If the underlying Σ protocol Σ is honest-verifier zero-knowledge and has high min-entropy commitments, then the transformed NIZK NIZK is zero-knowledge, that is, there are a set of simulators $(\mathsf{NIZK.SimSetup}, \mathsf{NIZK.SimRO}, \mathsf{NIZK.SimPrf})$ satisfying that for all PPT distinguisher $\mathcal{A}$, it holds that*

$$\begin{aligned} &|\Pr[\mathsf{crs} \leftarrow \Sigma.\mathsf{Setup}(1^n) : \mathcal{A}^{\mathcal{H},\mathcal{P}}(\mathsf{crs}) \to 1] \\ &\quad - \Pr[(\mathsf{crs}, \mathsf{state}_{\mathrm{RO}}) \leftarrow \mathsf{NIZK.SimSetup}(1^n) : \mathcal{A}^{\mathcal{O}_{\mathsf{NIZK.SimRO}}, \mathcal{O}_{\mathsf{NIZK.SimPrf}}}(\mathsf{crs}) \to 1]| \\ &\qquad\qquad = \mathsf{negl}(n) \end{aligned}$$

where the oracles are defined as follows.

$\mathcal{H}$. *This is a random oracle.*

$\mathcal{P}$. *This oracle, given a pair $(\mathsf{X}, \mathsf{W}) \in \mathcal{R}$, computes $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \mathsf{state}) \leftarrow \Sigma.\mathsf{Com}(\mathsf{crs}, \mathsf{X}, \mathsf{W})$, $\beta \leftarrow \Sigma.\mathsf{Chal}(\mathsf{crs}, \mathsf{X})$, and $\gamma \leftarrow \Sigma.\mathsf{Resp}(\mathsf{crs}, \mathsf{X}, \mathsf{W}, \beta, \mathsf{state})$, and returns $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma)$. If $(\mathsf{X}, \mathsf{W}) \notin \mathcal{R}$, this oracle returns $\bot$.*

$\mathcal{O}_{\mathsf{NIZK.SimRO}}$. *This oracle, given a random oracle query q, computes $(h, \mathsf{state}_{\mathrm{RO}}) \leftarrow \mathsf{NIZK.SimRO}(q, \mathsf{state}_{\mathrm{RO}})$, and returns h.*

$\mathcal{O}_{\mathsf{NIZK.SimPrf}}$. *This oracle, given a pair $(\mathsf{X}, \mathsf{W}) \in \mathcal{R}$, computes $((\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma), \mathsf{state}_{\mathrm{RO}}) \leftarrow \mathsf{NIZK.SimPrf}(\mathsf{X}, \mathsf{state}_{\mathrm{RO}})$ and returns $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma)$. If $(\mathsf{X}, \mathsf{W}) \notin \mathcal{R}$, this oracle returns $\bot$.*

Proof. We construct a set of simulators as follows.

The simulator $\mathsf{NIZK.SimSetup}(1^n)$ runs $(\mathsf{crs}, \mathsf{td}) \leftarrow \Sigma.\mathsf{SimSetup}(1^n)$ and store td and an empty hash list in $\mathsf{state}_{\mathrm{RO}}$. Then $\mathsf{NIZK.SimSetup}$ outputs $(\mathsf{crs}$, $maths f state_{\mathrm{RO}}$.

The $\mathsf{NIZK.SimRO}$, given a random oracle query q, uses the hash list stored in $\mathsf{state}_{\mathrm{RO}}$ to responds to the query. Namely, if the hash value of q is stored in the hash list, $\mathsf{NIZK.SimRO}$ responds with this hash value. Otherwise, $\mathsf{NIZK.SimRO}$ samples a fresh hash value β, store (q, β) in the hash list. Finally $\mathsf{NIZK.SimRO}$ outputs $(\beta, \mathsf{state}_{\mathrm{RO}})$ where $\mathsf{state}_{\mathrm{RO}}$ is the updated state information.

The simulator $\mathsf{NIZK.SimPrf}$, given a statement X and a label lbl, uses the trapdoor td stored in $\mathsf{state}_{\mathrm{RO}}$ and simulates a transcript $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma) \leftarrow \Sigma.\mathsf{Sim}(\mathsf{td}, \mathsf{X})$. Then $\mathsf{NIZK.SimPrf}$ generates a key pair of the one-time signature scheme $(\mathsf{vk}_{\mathrm{ot}}, \mathsf{sk}_{\mathrm{ot}}) \leftarrow \mathsf{SIG.KG}(1^n)$. After that, $\mathsf{NIZK.SimPrf}$ updates the hash list stored in $\mathsf{state}_{\mathrm{RO}}$ by programming the random oracle as $\beta = \mathcal{H}(\mathsf{crs}, \mathsf{X}, \mathsf{lbl}, \mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$ and generates a signature $\sigma_{\mathrm{ot}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{\mathrm{ot}}, \gamma)$. If the hash list already contain an entry for $(\mathsf{crs}, \mathsf{X}, \mathsf{lbl}, \mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$, $\mathsf{NIZK.SimPrf}$ outputs $(\perp, \mathsf{state}_{\mathrm{RO}})$. Otherwise, it outputs $((\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma), \mathsf{state}_{\mathrm{RO}})$.

This completes the description of the simulators.

To prove that NIZK is zero-knowledge with these simulators, we define the following games.

Game_0. The real game in the definition of the zero-knowledge property.

Game_1. In this game, the oracle $\mathcal{P}$ samples a transcript $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma)$ honestly and checks if $(\mathsf{crs}, \mathsf{X}, \mathsf{lbl}, \mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$ is already queried to the random oracle. If it is already queried, the oracle $\mathcal{P}$ returns $\perp$. Otherwise, $\mathcal{P}$ returns the honestly generated proof.

Game_2. In this game, a common reference string crs is sampled by running $(\mathsf{crs}, \mathsf{td}) \leftarrow \Sigma.\mathsf{SimSetup}(1^n)$. Furthermore, the oracle $\mathcal{P}$ samples a transcript by running $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma) \leftarrow \Sigma.\mathsf{Sim}(\mathsf{td}, \mathsf{X})$.

We claim that the adjacent games are indistinguishable. Let T_i be the event that the adversary $\mathcal{A}$ outputs 1.

We claim that $|\Pr[T_0] - \Pr[T_1]| = \mathsf{negl}(n)$. This is because the Σ protocol has high min-entropy commitments and thus $\mathcal{A}$ is unable to guess the freshly sampled transcripts $(\alpha_{\mathrm{unrec}}, \alpha_{\mathrm{rec}}, \beta, \gamma)$.

We claim that $|\Pr[T_1] - \Pr[T_2]| = \mathsf{negl}(n)$. This is due to the honest-verifier zero-knowledge property of the underlying Σ protocol.

To show this, we construct a reduction $\mathcal{B}$ attacking the zero-knowledge property of the Σ protocol. Let q be the number of proof queries which $\mathcal{A}$ makes during the experiment. Then, given a common reference string crs, $\mathcal{B}$ internally runs $\mathcal{A}$, simulating Game_1 or Game_2, with the exception that transcripts for simulating the oracle $\mathcal{P}$ are queried to $\mathcal{B}$'s own proof oracle. If $\mathcal{A}$ outputs $b \in \{0, 1\}$, then $\mathcal{B}$ outputs b.

This completes the definition of $\mathcal{B}$. Clearly, if $\mathcal{B}$ is given access to the real proof oracle, $\mathcal{B}$ simulates Game_1. Similarly, if $\mathcal{B}$ is given access to the simulated proof oracle, $\mathcal{B}$ simulates Game_2. Then, $|\Pr[T_1] - \Pr[T_2]|$ is equal to $\mathcal{B}$'s advantage in distinguishing real or simulated proofs. Thus this difference is negligible.

Finally, the game Game_2 is identical to the simulation game in the definition of zero-knowledge with the above simulators. Therefore, the above construction of simulators provides the zero-knowledge property.

Theorem F.2 (Simulation extractability). *If the underlying Σ protocol Σ is specially sound and the underlying signature scheme* SIG *is strongly unforgeable against chosen-message attacks, then the NIZK* NIZK *is simulation extractable with respect to the simulators* $(\mathsf{NIZK.SimSetup}, \mathsf{NIZK.SimRO}, \mathsf{NIZK.SimPrf})$ *defined in Theorem F.1 and with extraction error $\nu = Q/\nu + \mathsf{negl}(n)$ where Q is* *the number of hash queries made by $\mathcal{A}$ and $\mathsf{negl}(n)$ is a negligible function determined only by* SIG*, that is, for any PPT $\mathcal{A}$, there is an PPT extractor that satisfies the following. Let*

$$\mathsf{acc} = \Pr\left[\begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.SimSetup}(1^n), \\ \rho \leftarrow \{0,1\}^{\mathsf{poly}(n)}, \\ (\mathsf{X}^*, \mathsf{lbl}^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{RO}}, \mathcal{O}_{\mathrm{prf}}}(\mathsf{crs}; \rho) \\ : (\mathsf{X}^*, \mathsf{lbl}^*, \pi^*) \in \mathcal{T}_{\mathrm{prf}} \wedge \mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{lbl}^*, \mathsf{X}^*, \pi^*) = 1 \end{array}\right]$$

$$\mathsf{frk} = \Pr\left[\begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.SimSetup}(1^n), \\ \rho \leftarrow \{0,1\}^{\mathsf{poly}(n)}, \\ (\mathsf{X}^*, \mathsf{lbl}^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{RO}}, \mathcal{O}_{\mathrm{prf}}}(\mathsf{crs}; \rho) \\ \mathsf{W}^* \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, \mathsf{X}^*, \pi^*, \rho, \mathcal{T}_{\mathrm{RO}}, \mathcal{T}_{\mathrm{prf}}) \\ : (\mathsf{X}^*, \mathsf{lbl}^*, \pi^*) \notin \mathcal{T}_{\mathrm{prf}} \wedge \mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{lbl}^*, \mathsf{X}^*, \pi^*) = 1 \wedge (\mathsf{X}^*, \pi^*) \in \mathcal{R} \end{array}\right]$$

where the oracles $\mathcal{O}_{\mathrm{RO}}$ and $\mathcal{O}_{\mathrm{prf}}$ and the lists $\mathcal{T}_{\mathrm{RO}}$ and $\mathcal{T}_{\mathrm{prf}}$ are defined as follows.

$\mathcal{O}_{\mathrm{RO}}$. *The oracle $\mathcal{O}_{\mathrm{RO}}$, given a query q, returns h where $(h, \mathrm{state}_{\mathrm{RO}}) \leftarrow \mathsf{NIZK.SimRO}(q, \mathsf{state}_{\mathrm{RO}})$.*

$\mathcal{O}_{\mathrm{prf}}$. *The oracle $\mathcal{O}_{\mathrm{prf}}$, given a query X, returns π where $(\pi, \mathsf{state}_{\mathrm{RO}}) \leftarrow \mathsf{NIZK.SimPrf}(\mathsf{X}, \mathsf{lbl}, \mathsf{state}_{\mathrm{RO}})$.*

T_{RO}. *This is the set of the pairs of the queries to $\mathcal{O}_{\mathrm{RO}}$ and their responses.*

$\mathcal{T}_{\mathrm{prf}}$. *This is the set of the pairs of the queries to $\mathcal{O}_{\mathrm{prf}}$ and their responses.*

Then there exists a constant $d > 0$ and a polynomial p satisfying that whenever $\mathsf{acc} \geq \nu$, we have $\mathsf{ext} \geq (\mathsf{acc} - \nu)^d / p$.

Proof. Let $\mathcal{A}$ be an PPT adversary that outputs a proof.

Given $\mathcal{A}$, we construct an another PPT algorithm $\mathcal{A}'$ as follows. This algorithm $\mathcal{A}'$ is given as input a common reference string crs, a list of hash values $(\beta_1, \dots, \beta_Q)$, and a random tape ρ. Given this input, $\mathcal{A}'$ splits ρ into two random tapes into $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{A}'}$, and runs $\mathcal{A}(\mathsf{crs}; \rho_{\mathcal{A}})$. When $\mathcal{A}$ issues a hash query, $\mathcal{A}'$ consumes one element from $(\beta_1, \dots, \beta_Q)$ and returns it to $\mathcal{A}$. When $\mathcal{A}$ issues a proof query, $\mathcal{A}'$ use $\rho_{\mathcal{A}'}$ as randomness to simulate a proof as in $\mathsf{NIZK.SimPrf}$ in Theorem F.1. When $\mathcal{A}$ terminates with $(\mathsf{X}^*, \mathsf{lbl}^*, (\mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \beta^*, \gamma^*, \sigma^*_{\mathrm{ot}}))$, $\mathcal{A}'$ searches for the random oracle query $(\mathsf{crs}, \mathsf{X}^*, \mathsf{lbl}^*, \mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$ where $\bar{\alpha}^*_{\mathrm{rec}} \leftarrow \Sigma.\mathsf{Verify}(\mathsf{crs}, \mathsf{X}^*, \alpha^*_{\mathrm{unrec}}, \beta^*, \gamma^*)$ in the hash list. If such an entry is found, $\mathcal{A}'$ lets J be the index such that h_J was consumed to respond to this hash query. If such an index is found, $\mathcal{A}'$ outputs $(J, (h_J, \mathsf{X}^*, \Sigma^*_{\mathrm{unrec}}, \Sigma^*_{\mathrm{rec}}, \beta^*, \gamma^*))$. If no such an index is found, $\mathcal{A}'$ outputs $(0, \perp)$.

This completes the construction of $\mathcal{A}'$. Let acc be the probability in the theorem, E_{acc} be the event of that probability, and E_{idx} be the event that the index J is found. Then we have the following equality:

$$\mathsf{acc} = \Pr[E_{\mathrm{acc}}] = \Pr[E_{\mathrm{acc}} \wedge E_{\mathrm{idx}}] + \Pr[E_{\mathrm{acc}} \wedge \neg E_{\mathrm{idx}}].$$

We claim that $\Pr[E_{\mathrm{acc}} \wedge \neg E_{\mathrm{idx}}]$ is negligible, assuming that SIG is strongly unforgeable against chosen-message attack.

Lemma F.1. *If* SIG *is strongly unforgeable against chosen-message attacks, we have that* $\Pr[E_{\mathrm{acc}} \wedge \neg E_{\mathrm{idx}}]$ *is negligible.*

Proof. To this end, we construct a reduction $\mathcal{B}$ that attacks the strong unforgeability of SIG. The construction of $\mathcal{B}$ is as follows. Given $\mathsf{vk}_{\mathrm{ot}}$, $\mathcal{B}$ internally runs $\mathcal{A}$ and simulates the oracles as in $\mathcal{A}$ does with the following exceptions. Firstly, $\mathcal{B}$ chooses an index $k \leftarrow \{1, \ldots, Q_{\mathrm{prf}}\}$ randomly where Q_{prf} is the number of the proof queries that $\mathcal{A}$ makes. Secondly, for the k-th proof query, $\mathcal{B}$ uses $\mathsf{vk}_{\mathrm{ot}}$, which is given as input to $\mathcal{B}$ and obtains σ_{ot} by querying $\mathcal{B}$'s own signing oracle. Finally, when $\mathcal{A}$ terminates with output $(\mathsf{X}^*, \mathsf{lbl}^*, (\mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \beta^*, \gamma^*, \sigma^*_{\mathrm{ot}}))$, $\mathcal{B}$ outputs $(\gamma^*, \sigma^*_{\mathrm{ot}})$.

This completes the construction of $\mathcal{B}$. We claim that whenever $E_{\mathrm{acc}} \wedge \neg E_{\mathrm{idx}}$ occurs, $\mathcal{B}$ succeeds in forging a one-time signature with high probability. We claim that $\mathsf{vk}^*_{\mathrm{ot}} = \mathsf{vk}_{\mathrm{ot}}$ with probability $1/Q_{\mathrm{prf}}$. Since an index J is not found, the hash value $\mathcal{H}(\mathsf{crs}, \mathsf{X}^*, \mathsf{lbl}^*, \mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$ was programmed when responding to an proof query. Since $\mathcal{B}$ chooses randomly where it uses $\mathsf{vk}_{\mathrm{ot}}$ from all the proof queries, it holds that $\mathsf{vk}^*_{\mathrm{ot}} = \mathsf{vk}_{\mathrm{ot}}$ with probability $1/Q_{\mathrm{prf}}$. Then we claim that the output of $\mathcal{B}$ is a legitimate forgery of SIG. Remind that the output of $\mathcal{A}$ is different from any of the statement-label-proof triples which are $\mathcal{A}$'s proof query and its response. In other words, let $(\mathsf{X}, \mathsf{lbl}, (\mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \beta, \gamma, \sigma_{\mathrm{ot}}))$ be the query and its response such that the random oracle $\mathcal{H}(\mathsf{crs}, \mathsf{X}^*, \mathsf{lbl}^*, \mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$ was programmed when responding to this query. Then, since the inputs to $\mathcal{H}$ in the proof query and $\mathcal{A}$'s output are the same, we have $(\mathsf{X}^*, \mathsf{lbl}^*, \mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \bar{\alpha}^*_{\mathrm{rec}}) = (\mathsf{X}, \mathsf{lbl}, \mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}})$. Thus, we have that $\beta^* = \mathcal{H}(\mathsf{crs}, \mathsf{X}^*, \mathsf{lbl}^*, \mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \bar{\alpha}^*_{\mathrm{rec}}) = \mathcal{H}(\mathsf{crs}, \mathsf{X}, \mathsf{lbl}, \mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \bar{\alpha}_{\mathrm{rec}}) = \beta$. Due to the winning condition, we have that $(\mathsf{X}^*, \mathsf{lbl}^*, (\mathsf{vk}^*_{\mathrm{ot}}, \alpha^*_{\mathrm{unrec}}, \beta^*, \gamma^*, \sigma^*_{\mathrm{ot}})) \neq (\mathsf{X}, \mathsf{lbl}, (\mathsf{vk}_{\mathrm{ot}}, \alpha_{\mathrm{unrec}}, \beta, \gamma, \sigma_{\mathrm{ot}}))$. Therefore, we have that $(\gamma^*, \sigma^*_{\mathrm{ot}}) \neq (\gamma, \sigma_{\mathrm{ot}})$. Since the latter pair is $\mathcal{B}$'s signing query and its response, $\mathcal{B}$ succeeds in forging a one-time signature.

Now we have $\mathcal{A}'$ that outputs $(J, -)$ with $J \geq 1$ with probability $\mathsf{acc} - \Pr[E_{\mathrm{acc}} \wedge \neg E_{\mathrm{ind}}] = \mathsf{acc} - \mathsf{negl}(n)$. Applying the general forking lemma to this $\mathcal{A}'$, we have a rewinding algorithm $\mathcal{F}_{\mathcal{A}'}$, that outputs $(1, -, -)$ with probability greater than $(\mathsf{acc} - \mathsf{negl}(n))((\mathsf{acc} - \mathsf{negl}(n))/Q - 1/h)$. If $\mathcal{F}_{\mathcal{A}'}$ outputs $(1, (h_J, \mathsf{X}^*, \Sigma^*_{\mathrm{unrec}}, \Sigma^*_{\mathrm{rec}}, \beta^*, \gamma^*), (h'_J, \mathsf{X}^{**}, \Sigma^{**}_{\mathrm{unrec}}, \Sigma^{**}_{\mathrm{rec}}, \beta^{**}, \gamma^{**}))$, due to the construction, we have that $\Sigma.\mathsf{Verify}(\mathsf{crs}, \mathsf{X}^*, \alpha^*_{\mathrm{unrec}}, \beta^*, \gamma^*) = \Sigma.\mathsf{Verify}(\mathsf{crs}, \mathsf{X}^{**}, \alpha^{**}_{\mathrm{unrec}}, \beta^{**}, \gamma^{**}) \neq \bot$ and that $\mathsf{X}^* = \mathsf{X}^{**}$ and $\alpha^* = \alpha^{**}$. Therefore, due to the special soundness of Σ, we can extract a witness by running $\mathsf{W} \leftarrow \Sigma.\mathsf{Ext}(\mathsf{crs}, \mathsf{X}^*, \alpha^*_{\mathrm{unrec}}, \bar{\alpha}^*_{\mathrm{rec}}, \beta^*, \gamma^*, \beta^{**}, \gamma^{**})$ where $\bar{\alpha}^* \leftarrow \Sigma.\mathsf{Verify}(\mathsf{crs}, \mathsf{X}^*, \alpha^*_{\mathrm{unrec}}, \beta^*, \gamma^*)$.

Now if we let h be the size of the domain of the challenges of Σ, we have that

$$\begin{aligned}\mathsf{ext} &\geq (\mathsf{acc} - \mathsf{negl}(n)) \left(\frac{\mathsf{acc} - \mathsf{negl}(n)}{Q} - \frac{1}{h} \right) \\ &= \frac{1}{Q} \left((\mathsf{acc} - \mathsf{negl}(n))^2 - (\mathsf{acc} - \mathsf{negl}(n)) \frac{Q}{h} \right).\end{aligned}$$

Due to the theorem's assumption, we have that $\mathsf{acc} - Q/h - \mathsf{negl}(n) \geq 0$. This implies that $(\mathsf{acc} - \mathsf{negl}(n)) \cdot (Q/h) - (Q/h)^2 \geq 0$. Therefore,

$$\begin{aligned}
&\frac{1}{Q}\left((\mathsf{acc}-\mathsf{negl}(n))^2-(\mathsf{acc}-\mathsf{negl}(n))\frac{Q}{h}\right)\\
&\qquad\geq \frac{1}{Q}\left((\mathsf{acc}-\mathsf{negl}(n))^2-2(\mathsf{acc}-\mathsf{negl}(n))\frac{Q}{h}+\left(\frac{Q}{h}\right)^2\right)\\
&\qquad\qquad= \frac{1}{Q}\left(\mathsf{acc}-\mathsf{negl}(n)-\frac{Q}{h}\right)^2.
\end{aligned}$$

This concludes the proof.