Lattice-Based Techniques for Accountable Anonymity: Composition of Abstract Stern's Protocols and Weak PRF with Efficient Protocols from LWR

Rupeng Yang[1,2] ⋆, Man Ho Au[2] ⋆⋆, Junzuo Lai[3], Qiuliang Xu[1] ⋆⋆⋆, and Zuoxia Yu[2] †

[1] School of Computer Science and Technology, Shandong University, Jinan, 250101, China
orbbyrp@gmail.com, xql@sdu.edu.cn
[2] Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
csallen@comp.polyu.edu.hk, zuoxia.yu@gmail.com
[3] College of Information Science and Technology, Jinan University, Guangzhou, 510632, China
laijunzuo@gmail.com

Abstract. In an accountable anonymous system, a user is guaranteed anonymity and unlinkability unless some well-defined condition is met. A line of research focus on schemes that do not rely on any trusted third party capable of de-anonymising users. Notable examples include k-times anonymous authentication (k-TAA), blacklistable anonymous credentials (BLAC) and linkable ring signatures (LRS). All instances of these schemes are based on traditional number theoretic assumptions, which are vulnerable to quantum attacks.

One common feature of these schemes is the need to limit the number of times a key can be (mis-)used. Traditionally, it is usually achieved through the use of a pseudorandom function (PRF) which maps a user's key to a pseudonym, along with a proof of correctness. However, existing lattice-based PRFs do not interact well with zero-knowledge proofs. To bridge this gap, we propose and develop the following techniques and primitives:

- We formalize the notion of **weak PRF with efficient protocols**, which allows a prover to convince a verifier that the function F is evaluated correctly. Specifically, we provide an efficient construction based on the learning with rounding problem, which uses abstract Stern's Protocol to prove $y = \mathsf{F}_k(x)$ and $y \neq \mathsf{F}_k(x)$ without revealing x, y or k.
- We develop a general framework, which we call **extended abstract Stern's protocol**, to construct zero-knowledge arguments system for statements formed by conjunction and disjunction of sub-statements, who (or whose variants) are provable using abstract Stern's Protocol. Specifically, our system supports arbitrary monotonic propositions and allows a prover to argue polynomial relationships of the witnesses used in these sub-statements.

⋆ This work was mainly done when doing the internship at The Hong Kong Polytechnic University.
⋆⋆ Corresponding author.
⋆⋆⋆ Corresponding author.
† The second to the fifth authors are sorted in the alphabetical order.

As many existing lattice-based primitives also admit proofs using abstract Stern's protocol, our techniques can easily glue different primitives together for privacy-enhancing applications in a simple and clean way. Indeed, we propose three new schemes, all of which are the first of its kind, in the lattice setting. They also enjoy additional advantages over instances of the number-theoretic counterpart. Our k-TAA and BLAC schemes support concurrent enrollment while our LRS features logarithmic signature size without relying on a trusted setup. Our techniques enrich the arsenal of privacy-enhancing techniques and could be useful in the constructions of other schemes such as e-cash, unique group signatures, public key encryption with verifiable decryption, etc.

Keywords: Lattice-Based Cryptography, Zero-Knowledge Arguments of Knowledge, Privacy-Preserving Protocol, Accountable Anonymity

Table of Contents

1 Introduction

Anonymity is usually a favorable property when conducting authentication protocols. However, fully anonymity will lead to Deindividuation [Zim69,MB00], and consequentially misbehaviors ensue. For instance, Wikipedia has allowed users to edit content anonymously, but several anonymous users have misbehaved by posting inappropriate content, and ultimately, the Wikipedia block users attempting to edit contents with an open proxy [wik]. Therefore, in practice, we need ways to revoke anonymity. One approach, as have done in the group signature, revokable anonymous credential, etc., is to employ an opening authority to complete this task, but the anonymity of users in such systems is totally compromised if the opening authority is corrupted or broken. So solutions without relying on a trusted party are preferable.

In another line of research, Privacy-Preserving Authentication schemes with Accountable Anonymity (P^2A^3), which provide ways to detect, block or even trace and revoke users with misbehaviors without relying on a trusted party, and guarantee anonymity and unlinkability of users as long as they do not perform some well-defined misbehaviors, are constructed. This includes linkable ring signature [LWW04], *k*-times anonymous authentication [TFS04], blacklistable anonymous credential [TAKS07] etc. Generally, in a P^2A^3 scheme, the number of times of a key can be (mis-)used is limited. To count the number of times that a key has been (mis-)used, a *tag*, which will bind the key in an anonymous and unlinkable manner, is employed. In this way, the verifier can limit the number of times a user can use his key by limiting the number of bases that can generate tags, and can limit the number of times a user can misuse his key by requiring the user to prove that a limited number of tags labeled as misused are generated by him. To ensure that the introduction of the tag will not compromise the anonymity and the unlinkability of the user, the tag must be pseudorandom; to ensure that the tag can bind the key, no two keys are allowed to be mapped to the same tag. Besides, to guarantee that each user generates the correct tag, proof for the correct generation of the tag should also be provided; and to allow the user to prove that most tags are not generated by him, proof for the fact that "key does not map the tag" is needed. Apart from the tag, we may also need a signature scheme with efficient protocols (in this paper we call it "CL signature" for short) or an accumulator scheme to enable certification of users. Besides, as the statements needed to prove in P^2A^3 schemes are quite complex and often involve logic relations beyond conjunction, we also need a proof composition technique here.

One common problem of all current P^2A^3 schemes is that they are built on traditional number-theoretical assumptions, which are vulnerable to quantum attacks. Even worse, security of most of them schemes relies more or less on some non-standard assumptions, e.g. strong RSA assumption, q-Strong Diffie-Hellman assumption etc, which may bring potential security issues. Lattice-based cryptography seems promising to solve these problems. First, it is widely believed that lattice-based hard problems are resistant to quantum attacks. Then, commonly used problems, e.g. the Shortest Integer Solution (SIS) problem [Ajt96] and the Learning With Error (LWE) problem [Reg05], are as hard as some worst-case standard lattice problems. Besides, there are numerous works constructing lattice-based privacy-preserving authentication schemes currently. Especially, in recent works, lattice-based accumulator scheme [LLNW16] and lattice-

based CL signature scheme [LLM+16a], are constructed. But the remaining techniques and primitives are still missing in the lattice-based setting, thus we need:

- *A suitable function for generating the tag.* In current schemes, the tag is generated by mapping the secret key to a group element via deterministic functions, and the correct generation of the tag is proved by employing suitable techniques proving equations over group. We also need such a deterministic function admitting efficient proof in the lattice-based setting. To simplify the construction and security analysis, we hope to use this function as an abstract primitive. One may hope lattice-based pseudorandom function [BPR12] could work, but it is not known whether they admit an efficient proof since complex structures are introduced in their construction, also adaptively pseudorandomness seems overkill for our purpose. It is a folklore that such functions can be formalized as a Verifiable Random Function (VRF) [MRV99]. However, we observe that the VRF, and even the simulatable VRF [CL07], which has been used in constructing E-Cash [BCKL09], is not suitable for our purpose. The main reason is that they do not bind the secret key to the output of the function, which is crucial for enabling various security guarantees, e.g. non-frameability, authenticity, traceability etc. Moreover, some security properties, e.g. security without common reference string, which prevent simple and efficient constructions for VRF, are not needed in our applications. Besides, they do not provide a proper interface for the zero-knowledge proofs, which may bring inconvenience when used in practice. Therefore, we still need to define a suitable cryptographic primitive admitting efficient construction in the lattice-based setting to complete the task of tag generation.
- *A proof for the inequality in the lattice-based setting.* Current schemes use the proof of "inequality of discrete logarithms" [CS03] to prove "key does not map the tag". However, the idea used in [CS03] seems not applicable here due to lack of corresponding algebraic structure in the lattice-based setting. So, we need new approaches to complete this task.
- *A new general framework for combining lattice-based zero-knowledge proofs.* Current efficient frameworks supporting logic compositions of proofs [CDS94, DSDCPY94, CPS+16a, CPS+16b] seem unsuited for combining lattice-based proofs in practice. To see this, recall that when using those frameworks to construct proofs for compound statements, the resulting proof only guarantees that the prover knows correct witnesses for enough many sub-statements. However, in practical applications, it is usually required to additionally prove that different sub-statements share communal witnesses. In traditional number theoretical settings, this gap can be closed by committing these communal witnesses and additionally prove equivalent of committed values. But in the lattice-based settings, witnesses in different arguments may be (linearly) encoded via different methods. So, instead of proving that committed values are identical, we need to prove that committed values satisfy some linear equations. However, current lattice-based commitment schemes [KTX08, JKPT12, XXW13,BKLP15,BDOP16], though additively homomorphic, cannot support scalar multiplication of large values. Therefore, we need a new framework that can simultaneously prove logic compositions of sub-statements and linear relations of witnesses.

1.1 Our Results

In this paper, we bridge the gap between lattice-based assumptions and the P^2A^3 schemes via developing the following techniques and primitives.

- **A Lattice-Based Weak Pseudorandom Function with Efficient Protocols.** We define the weak PseudoRandom Function (wPRF) with efficient protocols. Here we write the function as $y = \mathsf{F}_k(x)$ for secret key k, input x and output y. The wPRF admits efficient ZKAoK proving that $y = \mathsf{F}_k(x)$ and that $y \neq \mathsf{F}_k(x)$. We also require the uniqueness of secret key, namely, for any output y and nearly any inputs x, there exists at most one secret key k satisfies $y = \mathsf{F}_k(x)$. We also construct it from lattice-based assumptions. Our new primitive is mainly used to generate public keys and "tags" when constructing P^2A^3 schemes in this work, and can be used to construct some other schemes, e.g. E-Cash, unique group signature etc. Especially, the ZKAoK proving $y \neq \mathsf{F}_k(x)$ is essential for enabling blacklist-based accountable anonymity, and is also applicable to construct some other schemes, e.g. public key encryption with verifiable decryption. Besides, we can also use it to explain how tags work in previous constructions of P^2A^3 schemes.
- **A General Framework for Combining Lattice-Based Zero-Knowledge Proofs.** We present a general framework to construct Zero-Knowledge Arguments of Knowledge (ZKAoK) for statements combined by a set of sub-statements, each of which is either of the form $((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathcal{L}_{\mathcal{R}}$ or of the form $((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \notin \mathcal{L}_{\mathcal{R}}$ for a relation $\mathcal{R}$ provable under the abstract Stern's protocol presented in [LLM+16a], and witnesses of these sub-statements satisfy some linear/polynomial equations. The framework could greatly reduce the complexity for constructing applications in this work (and relevant cryptographic primitives in future works), and enable the researchers to focus more on the ideas and less on the details of the constructed primitives. Another advantage of our general framework is that it provides a simple and efficient way to prove correct evaluation of arithmetic operations on secret values, thus is potentially useful in constructing applications such as blacklistable anonymous credential with reputation etc. The general framework contains three layers. First, we develop a compiler transforming a statement of the form $\boldsymbol{M} \cdot \boldsymbol{z} \neq \boldsymbol{u}$ into the form $\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v}$, where the latter is provable under the abstract Stern's protocol. Then, we present an argument for statements combined by sub-statements via monotone span programs, where each sub-statement is of the form $\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v}$, and witnesses of these sub-statements satisfy some linear equations. Finally, we also improve this argument to further support polynomial relations of witnesses.

Based on the wPRF and the general framework, we can construct a variety of P^2A^3 schemes, which are based on standard lattice problems with stronger security guarantees. Via the introduction of the general framework, these schemes can be constructed in a simple and clean way. In summary, the constructed applications include:

- A linkable ring signature scheme, which is also the first one that achieves a logarithmic signature size in the cardinality of the ring without relying on a trusted setup.
- A K-times anonymous authentication protocol, which is also the first one that supports concurrent enrollment.
- A blacklistable anonymous credential system, which is also the first one that supports concurrent enrollment.

1.2 Overview

The Composition of the abstract Stern's Protocols. To transform relations of the form $\boldsymbol{P} \cdot \boldsymbol{x} \neq \boldsymbol{v}$ into the form provable under abstract Stern's protocol, we adopt the technique presented in [LNSW13] to prove that the difference between $\boldsymbol{P} \cdot \boldsymbol{x}$ and $\boldsymbol{v}$ is not zero. Observe that the technique in [LNSW13] extends the witness in a specific manner when proving a vector is not zero, which makes it not applicable in some cases. To solve this problem, we present two new alternative techniques, which do not require a specific extension technique and are suitable for more scenarios. The first one is based on the fact that a vector is not zero iff its hamming weight is not zero, and introduces only an additional overhead logarithmic in the length of the vector, but cannot be used in the setting whose moduli is smaller than the length of the vector. The second technique relies on the fact that the inner product between a zero vector and any vector is zero, and does not require any additional condition, but will introduce an additional overhead linear in the length of the vector.

The general framework for combining lattice-based zero-knowledge proofs will transform a set of ZKAoKs following the abstract Stern's protocol, each of which proves statements of the form $((\boldsymbol{P}_i, \boldsymbol{v}_i), \boldsymbol{x}_i) \in \mathcal{R}_i$, into a new ZKAoK that follows the abstract Stern's protocol. The transformation works in three steps. In the first step, it generates a proof that can "prove" all sub-statement of a given statement. The main problem to complete this step is to "prove" the false sub-statements. In [CDS94], this problem is solved by assigning randomness for false sub-statements in advance to enable simulation proofs for them. Here in our setting, where each sub-statement is true iff it satisfies $\boldsymbol{P}_i \cdot \boldsymbol{x}_i = \boldsymbol{v}_i$, we can "prove" any sub-statement (in a fake manner) via proving that $\boldsymbol{P}_i \cdot (b_i \cdot \boldsymbol{x}_i) = b_i \cdot \boldsymbol{v}_i$, or alternatively $(\boldsymbol{P}_i, -\boldsymbol{v}_i) \cdot (b_i \cdot \boldsymbol{x}_i^\intercal, b_i)^\intercal = \boldsymbol{0}$, where b_i is 0 if the sub-statement is false. The output of the first step is a simple concatenation of all these fake proofs. In the second step, we need to additionally prove that set of indices of true statements can be accepted by the given monotone span program defined by a matrix $\boldsymbol{M}$ and a map ρ. Recall that this monotone span program accepts a set iff there exists vector $\boldsymbol{g}$ that $\boldsymbol{M}^\intercal \cdot \boldsymbol{g} = (1, 0, \ldots, 0)^\intercal$ and $\boldsymbol{g}[i] = 0$ if $\rho(i)$ is outside the set. So, it is sufficient to additionally prove that there exists $\boldsymbol{g}$ satisfying $\boldsymbol{M}^\intercal \cdot \boldsymbol{g} = (1, 0, \ldots, 0)^\intercal$ and $\boldsymbol{g}[i]$ is a multiple of $b_{\rho[i]}$. After these two steps, the resulting argument is still under the abstract Stern's protocol. So, we can plug linear equation indicating relations of witnesses into it easily, and that completes the task.

To upgrade the general framework to supporting polynomial relations of witnesses, the main problem is to prove the correctness of non-linear equations, namely equations of the form $z_0 = z_1 \cdot z_2 \ldots \cdot z_t$, where t is a positive integer, and for $i \in [0, t]$, z_i is one bit. One may hope to complete this task by extending the technique presented in [LLM+16b], which can prove that a bit is the product of two bits, to the higher degree cases directly. However, this will introduce an additional communication cost exponential in t. Here, we observe that as each variable in this non-linear equation is either 0 or 1, we can define $\ell = t - \sum_{i=1}^{t} z_i$ and reduce the task to proving that $(\ell \neq 0 \wedge z_0 = 0) \vee (\ell = 0 \wedge z_0 = 1)$. The new statement can be handled by our general framework and the technique of proving the inequality, and only a communication cost that is logarithmic in t is introduced.

Weak Pseudorandom Function with Efficient Protocols. Our wPRF with efficient protocols can be viewed as a variant of the VRF, but it additionally requires that for nearly all input output pair, there exists at most one secret key that can evaluate the input to the output. This property is very useful in constructing advanced schemes. To demonstrate this, we take our construction of linkable ring signature, where the secret key of a user is the secret key of the wPRF, and his public key and tag are outputs of the wPRF on some public random input, as an example. Also, we assume that for the used wPRF, there exist some "bad" keys that can be sampled with only a negligible probability, but for any input/output pair, one can efficiently find a "bad" key to explain it.[4] In this way, the adversary can obtain one more opportunity to sign via using the "bad" secret key mapped to his public key; he can even sign on behalf of a group of users without him via using the "bad" secret key mapped to the public key of a valid user; besides, he can frame an honest user via using the "bad" secret key mapped to her tag to sign one more message. Also, for sake of flexibility, supporting arguments are defined separately from the definition of the function. To make it versatile in practice, we require that the supporting arguments of the wPRF could work in different cases, including the basic case where only the secret key needs to be hidden, the case that both the secret key and the input need to be hidden, the case that both the secret key and the output needs to be hidden, and finally the case that the secret key, the input and the output need to be hidden. Besides, to enable simple and efficient constructions, similar to the definition of the simulatable VRF [CL07], we define wPRF with efficient protocols in the public parameters model; we also only require a weak pseudorandomness, which is also required in the weak VRF defined in [BGRV09].

Our construction of wPRF with efficient protocols is based on the Learning With Rounding (LWR) assumption[BPR12], which is as hard as the LWE assumption if suitable parameters are chosen. Recall that the LWR assumption states that $(\boldsymbol{A}, \lfloor p/q \cdot \boldsymbol{A} \cdot \boldsymbol{s} \rceil) \overset{c}{\approx} (\boldsymbol{A}, \boldsymbol{u})$ for uniformly chosen $\boldsymbol{A}$, $\boldsymbol{s}$ and $\boldsymbol{u}$. This immediately leads to a wPRF with secret key $\boldsymbol{s}$, input $\boldsymbol{A}$, and output $\lfloor p/q \cdot \boldsymbol{A} \cdot \boldsymbol{s} \rceil$. The weak pseudorandomness comes from the LWR assumption directly. To ensure that there exists at most one secret key that maps the input to the output for any output and for all but a negligible fraction of inputs, we require that the inputs are "tall" matrices. In this way, the products of an input and different secret keys distribute sparsely in the space and are not likely to be rounded to the same result after scaling. The ZKAoK proving the correct evaluation of the function is just arguments for the LWR relation (in different cases). The main problem to deal with LWR relations is that we have to handle equations of real numbers. Here, we solve this problem by requiring that q/p is an integer and transforming LWR relations into equivalent equations in $\mathbb{Z}_q$, which only consist of integers. The transformed equations are of the form $\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} = \boldsymbol{v}$, which looks identical to the LWE relation. However, in our proofs, we also need to guarantee the unique provability for the output of the function (even when the output is hidden). This is achieved by proving that each element in $\boldsymbol{v}$ is a multiple of q/p. In this way, there exists only one small enough $\boldsymbol{e}$ that satisfies the equation. Having guaranteed the two points, we can use the techniques in [LLM+16b]

[4] Such wPRF can be constructed easily. Let F be a secure wPRF, then we can construct a wPRF G satisfies the condition by defining $G_{k_1,k_2}(x)$ that outputs k_2 if all bits in k_1 are 0 and $F_{k_2}(x)$ otherwise, where k_1 is a long enough bit string.

to transform the statement into the relation provable under the abstract Stern's protocol. To prove the inequality of the output of the function and a given vector, we use the technique proving $\boldsymbol{M} \cdot \boldsymbol{z} \neq \boldsymbol{u}$ to transform the equation $\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} = \boldsymbol{v}$. One subtle issue here is to guarantee the correctness of $\boldsymbol{e}$. Here, we solve this issue by additionally proving that $\boldsymbol{e}$ can lead to a properly generated $\boldsymbol{v}$, namely multiple of q/p.

Applications. When constructing our applications, we will use the secret key of the wPRF with efficient protocols as the secret key of the application, and use the output on a public random input as the public key. Validity of a user is proved via proving that the public key is well certified. In the linkable ring signature scheme, this is done by proving that the user's public key is one of the public keys in the selected ring, and to accelerate the proof, we use the accumulator scheme in [LLNW16]. Note that public parameters of the used accumulator scheme can be sampled from a public random source, so our linkable ring signature scheme can achieve a logarithmic signature size without relying on a trusted setup. In the k-times anonymous authentication protocol and in the blacklistable anonymous credential system, public keys are signed by the group managers with the CL signature scheme in [LLM+16a], and in each authentication, the user needs to prove that he has the secret key for a properly signed public key. Note that in our construction, the group manager sign on the public key of the user, so we do not need the "Issue↔Obtain" protocol in these two constructions, and thus the constructed schemes can achieve the concurrent enrollment property. The tag is the output of the wPRF on some random inputs, where we guarantee the uniformity of each used input by letting it be the output of some cryptographic hash function, which will be modeled as a random oracle in the proof. To prove that no tags are generated by the user in the blacklistable anonymous credential system, we will use the proof of inequality for the LWR relation.

2 Preliminaries

Notations. In this paper, we will use bold lower-case letters (e.g. $\boldsymbol{v}$) to denote vectors, and use bold upper-case letters (e.g. $\boldsymbol{A}$) to denote matrices. All elements in vectors and matrices are integers unless otherwise specified. Also, we define function $M2V(\cdot)$ that takes as input a matrix $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$ and outputs a vector $\boldsymbol{a} = (\boldsymbol{A}[1]^\intercal \| \ldots \| \boldsymbol{A}[n]^\intercal)^\intercal$, where $\boldsymbol{A}[j]$ is the j-th column of the matrix $\boldsymbol{A}$ for $j \in [1, n]$. For a vector $\boldsymbol{v}$ of length n, we use $\|\boldsymbol{v}\|_\infty$ to denote the infinity norm of $\boldsymbol{v}$, and use $\|\boldsymbol{v}\|_1$ to denote the 1 norm of $\boldsymbol{v}$, and we also use $\boldsymbol{v}[i]$ to denote the ith element of $\boldsymbol{v}$ for $i \in [1, n]$. For a bit b, we use $\bar{b}$ to denote the negation of b.

Let $\mathcal{S}$ be a finite set, then we use $s \xleftarrow{\$} \mathcal{S}$ to denote sampling an element s uniformly from set $\mathcal{S}$. Also, for a distribution $\mathcal{D}$, we use $d \leftarrow \mathcal{D}$ to denote sampling d according to $\mathcal{D}$. We write $negl(\cdot)$ to denote a negligible function. For two random variables, say $\mathcal{X}$ and $\mathcal{Y}$, we write $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable. Let $\mathcal{R}$ be a binary relation, we use $\mathcal{L}_\mathcal{R}$ to denote the language characterized by $\mathcal{R}$.

Besides, in this paper, we will also use the monotone span program $\mathcal{M} = (\mathbb{F}, \boldsymbol{M}, \rho)$, where $\mathbb{F}$ is a field, $\boldsymbol{M} \in \mathbb{F}^{m \times n}$, ρ is a map from integers in $[1, m]$ to integers in $[1, m']$, and m, n, m' are poitive integers. The monotone span program can define an access structure

$\mathcal{A} \subseteq 2^{\{1,\ldots,m'\}}$ consisting of sets $\mathcal{S} \subseteq [1, m']$ that $(1, 0, 0, \ldots, 0)^\intercal \in \mathbb{F}^n$ is in the space spaned by rows of $\boldsymbol{M}_\mathcal{S}$, where $\boldsymbol{M}_\mathcal{S}$ is a submatrix of $\boldsymbol{M}$ that the ith row in $\boldsymbol{M}$ is also in $\boldsymbol{M}_\mathcal{S}$ iff $\rho(i) \in \mathcal{S}$ for $i \in [1, m]$.

2.1 Cryptographic Assumptions

We will build our wPRF with efficient protocols from the LWR assumption with unbounded polynomial samples, which is described as follows.

Definition 2.1 (LWR [BPR12]). *Let n be the security parameter, and moduli $q \geq p \geq 2$ be integers. For a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, define the LWR distribution $\mathcal{L}_{\boldsymbol{s}}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_p$ obtained by choosing a vector $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^n$, and outputting $(\boldsymbol{a}, b = \lfloor \boldsymbol{a}^\intercal \cdot \boldsymbol{s} \rceil_p)$. The decisional $\mathrm{LWR}_{n,q,p}$ problems is to distinguish between any desired number of independent samples $(\boldsymbol{a}_i, b_i) \xleftarrow{\$} \mathcal{L}_{\boldsymbol{s}}$ and the same number of samples drawn uniformly and independently from $\mathbb{Z}_q^n \times \mathbb{Z}_p$, where $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$. Here, as that in [BPR12], we define $\lfloor \boldsymbol{x} \rceil_p = \lfloor (p/q) \cdot \boldsymbol{x} \rceil$ in this paper.*

As proved in [BPR12], solving the decisional $\mathrm{LWR}_{n,q,p}$ problem is at least as hard as solving the decisional $\mathrm{LWE}_{n,q,\chi}$ problem (described in Definition 2.2 below) with an efficiently sampleable B-bound distribution χ over $\mathbb{Z}$, as long as $q \geq p \cdot B \cdot n^{\omega(1)}$, namely q is super-polynomially large. Subsequently, there are also some works [AKPW13, BGM+16, ASA16] considering how to improve the parameters in the reduction, however, all of them needs an a-priori bound on the number of samples, which makes it not suitable for constructing pseudorandom functions.

Definition 2.2 (LWE [Reg05]). *Let n be the security parameter, and moduli $q \geq 2$ be integers. Let χ be a distribution on $\mathbb{Z}$. For a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, define the LWE distribution $\mathcal{L}_{\boldsymbol{s}}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^n$, an integer $e \leftarrow \chi$, and outputting $(\boldsymbol{a}, b = \boldsymbol{a}^\intercal \cdot \boldsymbol{s} + e)$. The decisional $\mathrm{LWE}_{n,q,\chi}$ problems is to distinguish between any desired number of independent samples $(\boldsymbol{a}_i, b_i) \xleftarrow{\$} \mathcal{L}_{\boldsymbol{s}}$ and the same number of samples drawn uniformly and independently from $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$.*

We will also use the SIS assumption, which is described as follows.

Definition 2.3 (SIS [Ajt96]). *The $\mathrm{SIS}^\infty_{n,m,q,\beta}$ problem is to find a non-zero vector $\boldsymbol{x} \in \mathbb{Z}^m$ such that $\|\boldsymbol{x}\|_\infty \leq \beta$ and $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{0} \mod q$ given a matrix $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$.*

As summarized in [P+16], let $q \leq 2^{poly(n)}$, χ be (discretized) Gaussian error distribution of parameter $\alpha q \geq 2\sqrt{n}$, $\gamma = \tilde{O}(n/\alpha)$, where $0 < \alpha < 1$, then solving the decision-$\mathrm{LWE}_{n,q,\chi}$ problem is at least as hard as quantumly solving the GapSVP_γ and the SIVP_γ on arbitrary n-dimensional lattices; let $q \geq \beta\tilde{O}(\sqrt{n})$, $\gamma = \beta\tilde{O}(\sqrt{n})$, then solving the $\mathrm{SIS}^\infty_{n,m,q,\beta}$ problem is at least as hard as solving the GapSVP_γ and the SIVP_γ on arbitrary n-dimensional lattices.

2.2 Cryptographic Primitives

In this section, we review several cryptographic primitives used in this paper.

Zero-knowledge proof of knowledge. In a zero-knowledge proof of knowledge [GMR89] system, a prover proves to a verifier that he possesses the witness for a statement without revealing any additional information. When deployed in our applications, we will in fact use non-interactive zero-knowledge proof of knowledge, and require it to have security properties of completeness, soundness, zero-knowledge, extractability, simulation soundness and simulation extractability. To construct a proof system satisfying those properties, one can apply the well-known Fiat-Shamir heuristic [FS86] to transform a three round public coin interactive zero-knowledge proof of knowledge (sigma protocol) into a non-interactive one. One advantage led by the Fiat-Shamir transform is that the transformed non-interactive proof system can additionally admit a message as input, thus it is also called signature proof of knowledge (SPK), and is usually written as $SPK\{(w) : \mathcal{S}\}[m]$, where $\mathcal{S}$ is the statement to be proved, w is the witness, and m is the message. For convenience, the message m can be omitted if it is not explicitly indicated in context.

Accumulator. An accumulator scheme [BDM93] can accumulate a large set of inputs into a small value (the accumulator), and provide a short witness for each element in the accumulated set. Security of accumulator requires that no one can generate a valid witness for an element not in the accumulated set.

CL Signature Schemes. A CL signature scheme [CL01, CL02] is a signature scheme that allows a signer to sign on the commitment of a message, and allows one to prove the possession of a valid message signature pair in a zero-knowledge manner. Thus it can provide both the authentication and the privacy of message.

2.3 The Abstract Stern's Protocol

In [LLM+16a], an abstract Stern's Protocol for a large class of relations are proposed. More precisely, let $q \geq 2, D, L$ be postive integers, let VALID$\subseteq \{-1, 0, 1\}^L$, and let

$$\mathcal{R}_{abstract} = \{(\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x} \in \mathbb{Z}_q^{D\times L} \times \mathbb{Z}_q^D \times \text{VALID} : \boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q\}$$

Also, let $\mathcal{S}$ be a finite set and for each $\pi \in \mathcal{S}$, let $\mathtt{T}_\pi$ be a permutation on L elements. We say $(\mathcal{S}, \mathtt{T})$ is "valid" for $\mathcal{R}_{abstract}$ if they satisfy the following two conditions:

$$\begin{cases} \forall \pi \in \mathcal{S}, \boldsymbol{x} \in \text{VLAID} \Longleftrightarrow \mathtt{T}_\pi(\boldsymbol{x}) \in \text{VALID} \\ \forall \boldsymbol{x} \in \text{VALID}, \text{ if } \pi \xleftarrow{\$} \mathcal{S}, \text{ then } \mathtt{T}_\pi(\boldsymbol{x}) \text{ is uniformly distributed in VALID.} \end{cases} \tag{1}$$

Then, we have

Lemma 2.1 ([LLM+16a]). *Based on finite set $\mathcal{S}$, and permutation $\mathtt{T}$, which are valid for $\mathcal{R}_{abstract}$, one can construct statistical ZKAoK for the relation $\mathcal{R}_{abstract}$ with perfect completeness, soundness error 2/3, and communication cost $\tilde{O}(L \log q)$.*

Therefore, to employ the abstract Stern's protocol to prove a statement, one needs to first transform the statement into the form of the abstract relation $\mathcal{R}_{abstract}$, then specify a valid permutation family $(\mathcal{S}, \mathtt{T})$. Note that the second term in Condition (1) implies that for all vectors in VALID, the numbers of $-1, 0, 1$ must be constant, thus we

must also guarantee this after the transformation. Serveral tools, which are developed in recent works on Stern's protocol [LNSW13, LLNW14, ELL+15, LLNW16, LLM+16a, LLM+16b] can be used to complete these two tasks, and review these tools in Appendix A.

3 The Extended Abstract Stern's Protocol

3.1 The Overview

In this section, we present the extended abstract Stern's protocol that can combine abstract Stern's protocols.

Let q, D, L be positive integers, VALID$\subseteq \{-1, 0, 1\}^L$ be a set, and $\mathcal{R} = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D\times L} \times \mathbb{Z}_q^D \times \text{VALID} : \boldsymbol{P}\cdot\boldsymbol{x} = \boldsymbol{v} \mod q\}$ be an abstract Stern's relation. We define the "Inequality variant" of $\mathcal{R}$ as $\bar{\mathcal{R}} = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D\times L} \times \mathbb{Z}_q^D \times \text{VALID} : \boldsymbol{P}\cdot\boldsymbol{x} \neq \boldsymbol{v} \mod q\}$. Our extended abstract Stern's protocol aims to prove relations combined by multiple relations, each of which is either an abstract Stern's relation or its inequality variant.

More precisely, the combined relation $\mathcal{R}_{\mathfrak{A},\mathfrak{F}}$ can be defined by a family of access control policies $\mathfrak{A}$, which defines how the relations combine, and a family of functions $\mathfrak{F}$, which defines relations of witnesses, as following. Let t, k, q be positive integers. For $i \in [1, t]$, let D_i, L_i be positive integers, let $d_i \in \{0, 1\}$ be a bit, let $\text{VALID}_i \subseteq \{-1, 0, 1\}^{L_i}$ be a set, let $\mathcal{R}_{i,0} = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D_i\times L_i} \times \mathbb{Z}_q^{D_i} \times \text{VALID}_i : \boldsymbol{P}\cdot\boldsymbol{x} = \boldsymbol{v} \mod q\}$ be an abstract Stern's relation, let $\mathcal{R}_{i,1} = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D_i\times L_i} \times \mathbb{Z}_q^{D_i} \times \text{VALID}_i : \boldsymbol{P}\cdot\boldsymbol{x} \neq \boldsymbol{v} \mod q\}$ be the inequality variant of $\mathcal{R}_{i,0}$, and let $\mathcal{R}_i = \mathcal{R}_{i,d_i}$. Also, let $\mathsf{A} : 2^{\{1,2,\ldots,t\}} \to \{0, 1\}$ be an access control policy in $\mathfrak{A}$, $\mathsf{F} : \{-1, 0, 1\}^{\sum_{i=1}^t L_i} \to \mathbb{Z}_q^k$ be a function in $\mathfrak{F}$ and let $\boldsymbol{c} \in \mathbb{Z}_q^k$. Then, a tuple $((\boldsymbol{P}_1, \boldsymbol{v}_1, \ldots, \boldsymbol{P}_t, \boldsymbol{v}_t), (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)) \in \mathcal{R}_{\mathfrak{A},\mathfrak{F}}$ iff

$$\begin{cases} \mathcal{T} = \{i \mid i \in [1, t] \wedge ((\boldsymbol{P}_i, \boldsymbol{v}_i), \boldsymbol{x}_i) \in \mathcal{R}_i\} \\ \mathsf{A}(\mathcal{T}) = 1 \\ \mathsf{F}(\boldsymbol{x}_1^\intercal, \ldots, \boldsymbol{x}_t^\intercal) = \boldsymbol{c} \end{cases} \tag{2}$$

We construct the ZKAoK protocols for the new relation $\mathcal{R}_{\mathfrak{A},\mathfrak{F}}$ in two steps. First, in Sec. 3.2 we present a method that can transform an inequality variant $\bar{\mathcal{R}}$ of an abstract Stern's relation $\mathcal{R}$ into another abstract Stern's relation $\mathcal{R}'$. Moreover, given the abstract Stern's protocol for $\mathcal{R}$, we can construct the abstract Stern's protocol for $\mathcal{R}'$. By using this method, we can transform the relation $\mathcal{R}_{\mathfrak{A},\mathfrak{F}}$ into the relation $\mathcal{R}'_{\mathfrak{A},\mathfrak{F}}$, which is combined by abstract Stern's relations via the relation combiner $\mathfrak{A}$ and the witnesses combiner $\mathfrak{F}$. Then, in Sec. 3.3, we construct an abstract Stern's protocol for the relation $\mathcal{R}'_{\mathfrak{A},\mathfrak{F}}$ given the abstract Stern's protocols for the underlying abstract Stern's relations, when $\mathfrak{A}$ is the monotone span program and $\mathfrak{F}$ is the linear function. Therefore, we have

Theorem 3.1. *Let $\mathfrak{A}$ be the monotone span program, $\mathfrak{F}$ be the linear function, and $\mathcal{R}_{\mathfrak{A},\mathfrak{F}}$ defined as above. Then given abstract Stern's protocols proving relation $\mathcal{R}_{i,0}$ for $i \in [1, t]$, one can construct an abstrct Stern's protocol for $\mathcal{R}_{\mathfrak{A},\mathfrak{F}}$.*

We also construct in Appendix C.2 an abstract Stern's protocol for the relation $\mathcal{R}'_{\mathfrak{A},\mathfrak{F}}$ given the abstract Stern's protocols for the underlying abstract Stern's relations, when $\mathfrak{A}$ is the monotone span program and $\mathfrak{F}$ is the mulivariate polynomial with binary input. As a result, we have

Theorem 3.2. *Let $\mathfrak{A}$ be the monotone span program, $\mathfrak{F}$ be the mulivariate polynomial with binary input, and $\mathcal{R}_{\mathfrak{A},\mathfrak{F}}$ defined as above. Then given abstract Stern's protocols proving relation $\mathcal{R}_{i,0}$ for $i \in [1,t]$, one can construct an abstrct Stern's protocol for $\mathcal{R}_{\mathfrak{A},\mathfrak{F}}$.*

3.2 Protocols for The Inequality Variant of An Abstract Stern's Relation

In this section, we present an abstract Stern's protocol proving the inequality variant of an abstract Stern's relation.

More precisely, let D', L', q be positive integers, let $\mathrm{VALID}' \subseteq \{-1, 0, 1\}^{L'}$ be a set, and let $\mathcal{R}' = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D' \times L'} \times \mathbb{Z}_q^{D'} \times \mathrm{VALID}' : \boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q\}$ be an abstract Stern's relation. Then, given common input $\boldsymbol{P'} \in \mathbb{Z}_q^{D' \times L'}$, $\boldsymbol{v'} \in \mathbb{Z}_q^{D'}$, and prover's secret input $\boldsymbol{x'} \in \mathrm{VALID}'$, the prover needs to prove that he knows a vector $\boldsymbol{x'} \in \mathrm{VALID}'$ that $\boldsymbol{P'} \cdot \boldsymbol{x'} \neq \boldsymbol{v'} \mod q$.

Now, let $\boldsymbol{u} = \boldsymbol{P'} \cdot \boldsymbol{x'} \mod q$, and $\boldsymbol{u'} = \boldsymbol{v'} - \boldsymbol{u} \mod q$, then it is sufficient to prove that

$$\begin{cases} \boldsymbol{P'} \cdot \boldsymbol{x'} - \boldsymbol{u} = \boldsymbol{0} \mod q \\ \boldsymbol{u} + \boldsymbol{u'} = \boldsymbol{v'} \mod q \\ \boldsymbol{u'} \neq \boldsymbol{0} \mod q \end{cases} \tag{3}$$

Here, the third term in Equation (3) can be handled by the method proposed by Ling et.al in [LNSW13], which is used to prove the SIS relation. With this method, to prove that $\boldsymbol{u'}$ is not equal to $\boldsymbol{0}$, we need to extend the decomposition of $\boldsymbol{u'}$ in a special manner, namely, extending the vector with one less 0. [5] In more detail, let $\dot{D} = D'\delta_{q-1}$, then we define $\tilde{\boldsymbol{u}}' = Dec_{D',q-1}(\boldsymbol{u'})$. Assume that there are n_0 0s and n_1 1s in $\tilde{\boldsymbol{u}}'$, then we append $\dot{D} - n_0 - 1$ 0s and $\dot{D} - n_1$ 1s to $\tilde{\boldsymbol{u}}'$ and denote the resulting vector, which has $\dot{D} - 1$ 0s and $\dot{D}$ 1s, as $\hat{\boldsymbol{u}}'$. We also define $\hat{\boldsymbol{K}}'_{D',q-1,2} = (\boldsymbol{K}_{D',q-1} \| 0^{m \times \dot{D}-1})$, which satisfies $\boldsymbol{u'} = \hat{\boldsymbol{K}}' \cdot \hat{\boldsymbol{u}}'$. Note that $\hat{\boldsymbol{u}}'$ can only be generated when $\tilde{\boldsymbol{u}}'$ is not $\boldsymbol{0}$, i.e. when $\boldsymbol{u'}$ is not $\boldsymbol{0}$. Then, we decompose and extend the remaining part of Equation (3) and define the following matrices and vectors.

$$\begin{cases} \hat{\boldsymbol{u}} = DecEnc2_{D',q-1}(\boldsymbol{u}), \boldsymbol{x} = (\boldsymbol{x'}^{\intercal} \| \hat{\boldsymbol{u}}^{\intercal} \| \hat{\boldsymbol{u}}'^{\intercal})^{\intercal} \\ \boldsymbol{P} = \begin{pmatrix} \boldsymbol{P'} & -\hat{\boldsymbol{J}}_{D',q-1,2} & 0^{D' \times 2\dot{D}-1} \\ 0^{D' \times L'} & \hat{\boldsymbol{J}}_{D',q-1,2} & \hat{\boldsymbol{K}}'_{D',q-1,2} \end{pmatrix} \\ \boldsymbol{v} = ((0^{D'})^{\intercal} \| \boldsymbol{v'}^{\intercal})^{\intercal} \end{cases}$$

[5] Since one needs to extend the target vector in a specific manner when using the method, it may be inconvenient to use this method in some cases. So, we also present an alternative method proving that a vector is not 0 in Sec. C.1, which may be of independent interest.

Then, we can transform Equation (3) as follows,

$$\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q$$

and define VALID= $\text{VALID}' \times \mathcal{D}_{\dot{D}} \times \{\boldsymbol{w} \in \{0,1\}^{2\dot{D}-1} \mid \|\boldsymbol{w}\|_1 = \dot{D}\}$ that is a set of vectors of length $L = L' + 4\dot{D} - 1$. Obviously, $\boldsymbol{x} \in$VALID. It can be easily verified that this statement is equivalent to the original statement.

Then we need to specify the permutation used in the protocol. As the relation $\mathcal{R}'$ can be proved under the abstract Stern's protocol, there exists $\mathcal{S}', \mathtt{T}'$ that are valid for $\mathcal{R}'$. Then we define $\mathcal{S} = \mathcal{S}' \times \{0,1\}^{\dot{D}} \times \mathcal{P}_{2\dot{D}-1}$. For $\pi = (\pi_1, \boldsymbol{b}_2, \pi_3) \in \mathcal{S}$, where $\pi_1 \in \mathcal{S}'$, and $\boldsymbol{b}_2 \in \{0,1\}^{\dot{D}}$ and $\pi_3 \in \mathcal{P}_{2\dot{D}-1}$, and $\boldsymbol{w} = (\boldsymbol{w}_1^\intercal \| \boldsymbol{w}_2^\intercal \| \boldsymbol{w}_3^\intercal)^\intercal$ where $\boldsymbol{w}_1$ is of length L', $\boldsymbol{w}_2$ is of length $2\dot{D}$ and $\boldsymbol{w}_3$ is of length $2\dot{D}-1$, we define $\mathtt{T}_\pi(\boldsymbol{w}) = (\mathtt{T}'_{\pi_1}(\boldsymbol{w}_1)^\intercal, \mathtt{F}_{\boldsymbol{b}_2}(\boldsymbol{w}_2)^\intercal, \pi_3(\boldsymbol{w}_3)^\intercal)^\intercal$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

3.3 Protocols for The Monotone Span Program Composition of Abstract Stern's Relations and Linear Equations of Witnesses

In this section, we present an abstract Stern's protocol for the relation combined by abstract Stern's relations via a monotone span program and a linear function, i.e. the protocol proves that the indices of true statements in a set of statements, each of which is about an abstract Stern's relation, satisfy a monotone span program, and witnesses of these statements satisfy a linear equation.

More precisely, let t, q, k be positive integers, q' be a positive prime and $\tilde{q} = \text{lcm}(q, q')$; let $\gamma = \tilde{q}/q$ and $\gamma' = \tilde{q}/q'$; let $\mathcal{M} = (\mathbb{Z}_{q'}, \boldsymbol{M}, \rho)$ be a monotone span program accepting an access structure $\mathcal{A} \subseteq 2^{\{1,\ldots,t\}}$, where $\boldsymbol{M} \in \mathbb{Z}_{q'}^{\ell_1 \times \ell_2}$; for $i \in [1,t]$, let D_i, L_i be positive integers, let $\text{VALID}_i \subseteq \{-1,0,1\}^{L_i}$ be a set, and let $\mathcal{R}_i = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D_i \times L_i} \times \mathbb{Z}_q^{D_i} \times \text{VALID}_i : \boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q\}$ be an abstract Stern's relation; let $\boldsymbol{B} \in \mathbb{Z}_q^{k \times (\sum_{i=1}^t L_i)}$ and $\boldsymbol{c} \in \mathbb{Z}_q^k$. Then, given common input $\boldsymbol{P}_i \in \mathbb{Z}_q^{D_i \times L_i}$, $\boldsymbol{v}_i \in \mathbb{Z}_q^{D_i}$, and prover's secret input $\boldsymbol{x}_i \in \text{VALID}_i$ for $i \in [1,t]$, the prover needs to prove that

$$\begin{cases} \mathcal{T} = \{i \mid i \in [1,t] \wedge ((\boldsymbol{P}_i, \boldsymbol{v}_i), \boldsymbol{x}_i) \in \mathcal{R}_i\} \\ \mathcal{T} \in \mathcal{A} \\ \boldsymbol{B} \cdot (\boldsymbol{x}_1^\intercal \| \ldots \| \boldsymbol{x}_t^\intercal)^\intercal = \boldsymbol{c} \mod q \end{cases} \tag{4}$$

Now, let b_i be a bit that equals to 1 iff $i \in \mathcal{T}$ for $i \in [1,t]$. Note that if $b_i = 1$, then $\boldsymbol{P}_i \cdot \boldsymbol{x}_i = \boldsymbol{v}_i$, thus we have $b_i \cdot (\boldsymbol{P}_i \cdot \boldsymbol{x}_i - \boldsymbol{v}_i) = 0$ for $i \in [1,t]$. Also, let $\boldsymbol{g} = (g_1, \ldots, g_{\ell_1})^\intercal \in \mathbb{Z}_{q'}^{\ell_1}$ be a vector that $\boldsymbol{g}^\intercal \cdot \boldsymbol{M} = (1,0,\ldots,0) \mod q'$ and $g_i = 0$ if $b_{\rho(i)} = 0$ for $i \in [1,\ell_1]$.[6] Then we rewrite Equation (4) as follows. Note that to handle equations under different modulars, we employ the technique in [YAY17].

[6] Since $\mathcal{T}$ satisfy $\mathcal{M}$, we have that rows of $\boldsymbol{M}$ that are mapped by ρ to those is that $b_i = 1$, span the row vector $(1,0,\ldots,0)$. Thus, one can compute a valid $\boldsymbol{g}$ efficiently by solving linear equations.

$$\begin{pmatrix} \gamma \cdot \hat{\boldsymbol{P}} & -\gamma \cdot \hat{\boldsymbol{V}} & & \\ & & \gamma \cdot \boldsymbol{B} & \\ & & & \gamma' \cdot \boldsymbol{M}^{\intercal} \end{pmatrix} \cdot \begin{pmatrix} b_1 \cdot \boldsymbol{x}_1 \\ \vdots \\ b_t \cdot \boldsymbol{x}_t \\ b_1 \\ \vdots \\ b_t \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_t \\ \boldsymbol{g} \end{pmatrix} = \begin{pmatrix} 0^{\hat{D}} \\ \gamma \cdot \boldsymbol{c} \\ \gamma' \cdot \boldsymbol{e} \end{pmatrix} \mod \tilde{q} \tag{5}$$

where in this section, we define $\hat{D} = \sum_{i=1}^{t} D_i$, $\hat{L} = \sum_{i=1}^{t} L_i$, $\boldsymbol{e} = (1, 0, \ldots, 0)^{\intercal} \in \{0,1\}^{\ell_2}$, and define

$$\hat{\boldsymbol{P}} = \begin{pmatrix} \boldsymbol{P}_1 & & \\ & \ddots & \\ & & \boldsymbol{P}_t \end{pmatrix} \qquad \hat{\boldsymbol{V}} = \begin{pmatrix} \boldsymbol{v}_1 & & \\ & \ddots & \\ & & \boldsymbol{v}_t \end{pmatrix}$$

Next, we need to decompose and extend Equation (5), and we first define the following matrices and vectors.

$$\begin{cases} \acute{\boldsymbol{x}} = (b_1 \cdot \boldsymbol{x}_1^{\intercal} \| \ldots \| b_t \cdot \boldsymbol{x}_t^{\intercal})^{\intercal}, \grave{\boldsymbol{x}} = (\bar{b}_1 \cdot \boldsymbol{x}_1^{\intercal} \| \ldots \| \bar{b}_t \cdot \boldsymbol{x}_t^{\intercal})^{\intercal} \\ \acute{\boldsymbol{b}} = (b_1, \ldots, b_t)^{\intercal}, \grave{\boldsymbol{b}} = (\bar{b}_1, \ldots, \bar{b}_t)^{\intercal} \\ \forall i \in [1, \ell_1], \hat{\boldsymbol{g}}_i = DecEnc2_{1,q'-1}(\boldsymbol{g}[i]) \\ \acute{\boldsymbol{g}} = (b_{\rho(1)} \cdot \hat{\boldsymbol{g}}_1^{\intercal} \| \ldots \| b_{\rho(\ell_1)} \cdot \hat{\boldsymbol{g}}_{\ell_1}^{\intercal})^{\intercal}, \grave{\boldsymbol{g}} = (\bar{b}_{\rho(1)} \cdot \hat{\boldsymbol{g}}_1^{\intercal} \| \ldots \| \bar{b}_{\rho(\ell_1)} \cdot \hat{\boldsymbol{g}}_{\ell_1}^{\intercal})^{\intercal} \\ \boldsymbol{x} = (\acute{\boldsymbol{x}}^{\intercal} \| \grave{\boldsymbol{x}}^{\intercal} \| \acute{\boldsymbol{b}}^{\intercal} \| \grave{\boldsymbol{b}}^{\intercal} \| \acute{\boldsymbol{g}}^{\intercal} \| \grave{\boldsymbol{g}}^{\intercal})^{\intercal} \\ \\ \boldsymbol{P} = \begin{pmatrix} \gamma \cdot \hat{\boldsymbol{P}} & 0^{\hat{D} \times \hat{L}} & -\gamma \cdot \hat{\boldsymbol{V}} & 0^{\hat{D} \times t} & 0^{\hat{D} \times 2\ell_1 \delta_{q'-1}} & 0^{\hat{D} \times 2\ell_1 \delta_{q'-1}} \\ \gamma \cdot \boldsymbol{B} & \gamma \cdot \boldsymbol{B} & 0^{k \times t} & 0^{k \times t} & 0^{k \times 2\ell_1 \delta_{q'-1}} & 0^{k \times 2\ell_1 \delta_{q'-1}} \\ 0^{\ell_2 \times \hat{L}} & 0^{\ell_2 \times \hat{L}} & 0^{\ell_2 \times t} & 0^{\ell_2 \times t} & \gamma' \cdot \boldsymbol{M}^{\intercal} \cdot \hat{\boldsymbol{J}}_{\ell_1, q'-1, 2} & 0^{\ell_2 \times 2\ell_1 \delta_{q'-1}} \end{pmatrix} \\ \\ \boldsymbol{v} = ((0^{\hat{D}})^{\intercal} \| \gamma \cdot \boldsymbol{c}^{\intercal} \| \gamma' \cdot \boldsymbol{e}^{\intercal})^{\intercal} \end{cases}$$

Then, we can transform Equation (5) as follows,

$$\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod \tilde{q}$$

and define

$$\begin{aligned}\text{VALID} = \{&(\boldsymbol{w}_{1,1}^\intercal \| \ldots \| \boldsymbol{w}_{1,t}^\intercal \| \boldsymbol{w}_{2,1}^\intercal \| \ldots \| \boldsymbol{w}_{2,t}^\intercal \| \boldsymbol{w}_{3,1}^\intercal \| \ldots \| \boldsymbol{w}_{3,t}^\intercal \| \\ &\boldsymbol{w}_{4,1}^\intercal \| \ldots \| \boldsymbol{w}_{4,t}^\intercal \| \boldsymbol{w}_{5,1}^\intercal \| \ldots \| \boldsymbol{w}_{5,\ell_1}^\intercal \| \boldsymbol{w}_{6,1}^\intercal \| \ldots \| \boldsymbol{w}_{6,\ell_1}^\intercal)^\intercal \mid \\ &\forall i \in [1,t], \forall j \in [1,\ell_1] \text{ that } \rho(j) = i, \\ &(\boldsymbol{w}_{1,i} \in \text{VALID}_i \wedge \boldsymbol{w}_{2,i} \in 0^{L_i} \wedge \boldsymbol{w}_{3,i} = 1 \wedge \boldsymbol{w}_{4,i} = 0 \wedge \\ &\boldsymbol{w}_{5,j} \in \mathcal{D}_{\delta_{q'-1}} \wedge \boldsymbol{w}_{6,j} \in 0^{2\delta_{q'-1}}) \vee \\ &(\boldsymbol{w}_{1,i} \in 0^{L_i} \wedge \boldsymbol{w}_{2,i} \in \text{VALID}_i \wedge \boldsymbol{w}_{3,i} = 0 \wedge \boldsymbol{w}_{4,i} = 1 \wedge \\ &\boldsymbol{w}_{5,j} \in 0^{2\delta_{q'-1}} \wedge \boldsymbol{w}_{6,j} \in \mathcal{D}_{\delta_{q'-1}})\}\end{aligned}$$

that is a set of vectors of length $L = 2\hat{L} + 2t + 4\ell_1\delta_{q'-1}$. Obviously, $\boldsymbol{x} \in$VALID. Note that for any $i \in [1,t]$, we have $b_i \cdot \boldsymbol{x}_i + \bar{b}_i \cdot \boldsymbol{x}_i = \boldsymbol{x}_i$, thus here we do not need to include $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$ explicitly in the witness. Besides, since $\boldsymbol{g}[j] = 0$ if $b_{\rho(j)} = 0$ for $j \in [1,\ell_1]$, we have $\hat{\boldsymbol{J}}_{\ell_1,q'-1,2} \cdot \hat{\boldsymbol{g}} = \hat{\boldsymbol{J}}_{\ell_1,q'-1,2} \cdot \hat{\boldsymbol{g}} = \boldsymbol{g}$, where we use $\hat{\boldsymbol{g}}$ to denote $(\hat{\boldsymbol{g}}_1^\intercal \| \ldots \| \hat{\boldsymbol{g}}_{\ell_1}^\intercal)^\intercal$ here. It can be easily verified that this statement is equivalent to the original statement indicated by Equation (4).

Then we need to specify the permutation used in the protocol. As each relation $\mathcal{R}_i$ can be proved under the abstract Stern's protocol, there exists $\mathcal{S}_i, \mathtt{T}^{(i)}$ that are valid for $\mathcal{R}_i$ for $i \in [1,t]$. Then we define $\mathcal{S} = \{0,1\}^t \times \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_t \times (\{0,1\}^{\delta_{q'-1}})^{\ell_1}$. For $\pi = (\boldsymbol{b}_0, \pi_1, \pi_2, \ldots, \pi_t, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_1}) \in \mathcal{S}$, where $\boldsymbol{b}_0 \in \{0,1\}^t$, $\pi_i \in \mathcal{S}_i$ for $i \in [1,t]$, and $\boldsymbol{b}_j \in \{0,1\}^{\delta_{q'-1}}$ for $j \in [1,\ell_1]$, and $\boldsymbol{w} = (\boldsymbol{w}_{1,1}^\intercal, \ldots, \boldsymbol{w}_{1,t}^\intercal, \boldsymbol{w}_{2,1}^\intercal, \ldots, \boldsymbol{w}_{2,t}^\intercal, \boldsymbol{w}_{3,1}^\intercal, \ldots, \boldsymbol{w}_{3,t}^\intercal, \boldsymbol{w}_{4,1}^\intercal, \ldots, \boldsymbol{w}_{4,t}^\intercal, \boldsymbol{w}_{5,1}^\intercal, \ldots, \boldsymbol{w}_{5,\ell_1}^\intercal, \boldsymbol{w}_{6,1}^\intercal, \ldots, \boldsymbol{w}_{6,\ell_1}^\intercal)^\intercal$ where for $i \in [1,t]$, $\boldsymbol{w}_{1,i}, \boldsymbol{w}_{2,i}$ is of length L_i, $\boldsymbol{w}_{3,i}, \boldsymbol{w}_{4,i}$ is of length 1, and for $j \in [1,\ell_1]$ $\boldsymbol{w}_{5,j}, \boldsymbol{w}_{6,j}$ is of length $2\delta_{q'-1}$, we define $\mathtt{T}_\pi(\boldsymbol{w}) = (\boldsymbol{w'}_{1,1}^\intercal, \ldots, \boldsymbol{w'}_{1,t}^\intercal, \boldsymbol{w'}_{2,1}^\intercal, \ldots, \boldsymbol{w'}_{2,t}^\intercal, \boldsymbol{w'}_{3,1}^\intercal, \ldots, \boldsymbol{w'}_{3,t}^\intercal, \boldsymbol{w'}_{4,1}^\intercal, \ldots, \boldsymbol{w'}_{4,t}^\intercal, \boldsymbol{w'}_{5,1}^\intercal, \ldots, \boldsymbol{w'}_{5,\ell_1}^\intercal, \boldsymbol{w'}_{6,1}^\intercal, \ldots, \boldsymbol{w'}_{6,\ell_1}^\intercal)^\intercal$ that for $i \in [1,t]$, $\boldsymbol{w'}_{1,i} = \mathtt{T}_{\pi_i}^{(i)}(\boldsymbol{w}_{1+\boldsymbol{b}_0[i],i})$, $\boldsymbol{w'}_{2,i} = \mathtt{T}_{\pi_i}^{(i)}(\boldsymbol{w}_{2-\boldsymbol{b}_0[i],i})$, $\boldsymbol{w'}_{3,i} = \boldsymbol{w}_{3+\boldsymbol{b}_0[i],i}$, $\boldsymbol{w'}_{4,i} = \boldsymbol{w}_{4-\boldsymbol{b}_0[i],i}$, and for $j \in [1,\ell_1]$, $\boldsymbol{w'}_{5,j} = \mathtt{F}_{\boldsymbol{b}_j}(\boldsymbol{w}_{5+\boldsymbol{b}_0[\rho(j)],j})$, $\boldsymbol{w'}_{6,j} = \mathtt{F}_{\boldsymbol{b}_j}(\boldsymbol{w}_{6-\boldsymbol{b}_0[\rho(j)],j})$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

4 Weak Pseudorandom Function with Efficient Protocols

In this section, we define and construct weak pseudorandom function with efficient protocols, which can be used to generate tags when constructing P^3A^2 schemes. Compared to a normal wPRF, the new primitive admits a series of ZKAoK protocols proving the correct evaluation of the function in different scenarios. Besides, it also has the "uniqueness" property, which can bind the secret key of the function to its output. To construct wPRF with efficient protocols, we start with the wPRF constructed implicitly in [BPR12]. Then we adapt the parameters of their consturction to achieve (different levels) of uniqueness. To complete the construction, we also develop a series of ZKAoKs proving the correct evaluation of the function under the abstract Stern's protocol.

4.1 The Definition

Formally, a wPRF with efficient protocols consists of two algorithms:

– $sk \leftarrow KeyGen(1^\lambda)$. On input a security parameter 1^λ, the key generation algorithm outputs the secret key sk for the function.
– $y = Eval(sk, x)$. On input a secret key sk and an input x, the evaluation algorithm outputs the output y of the function. Usually, we write this procedure as $y = F_{sk}(x)$.

Besides, it also consists of following ZKAoK protocols:

– A set of protocols proving $y = F_{sk}(x)$ with either hidden sk, or hidden (sk, x), or hidden (sk, y), or hidden (sk, x, y).
– A set of protocols proving $y \neq F_{sk}(x)$ with either hidden sk, or hidden (sk, x), or hidden (sk, y), or hidden (sk, x, y).

We require that a wPRF with efficient protocols has the following properties:

– **Weak Pseudorandomness.** Let $sk \leftarrow KeyGen(1^\lambda)$; let $O_{0,sk}()$ be an oracle that answers each query with a tuple (x, y) where x is sampled freshly and uniformly from the domain of the function, and $y = Eval(sk, x)$; let $O_{1,sk}()$ be an oracle that answers each query with a tuple (x, y) where x, y are sampled freshly and uniformly from the domain of the function and the range of the function respectively. Then for any probabilistic polynomial time adversary $\mathcal{A}$, $\Pr[b \xleftarrow{\$} \{0,1\}; b' \leftarrow \mathcal{A}^{O_{b,sk}}(); b = b'] \leq 1/2 + negl(\lambda)$.
– **Uniqueness.** Let x be an input sampled uniformly at random from the domain of the function, then $\Pr[\exists sk_1, sk_2, sk_1 \neq sk_2 \wedge F_{sk_1}(x) = F_{sk_2}(x)] \leq negl(\lambda)$, where the probability takes over the randomly chosen of the input x.

The uniqueness requires that for all but a negligible fraction of inputs, the output can bind the secret key when fixing the input. In some scenarios, a stronger version of the uniqueness, which requires that the output bounds the secret key even without fixing the input, is needed, and we define this stronger uniqueness as follows:
– **Strong Uniqueness.** Let x_1, x_2 be two inputs sampled independently and uniformly at random from the domain of the function, then $\Pr[\exists sk_1, sk_2, sk_1 \neq sk_2 \wedge F_{sk_1}(x_1) = F_{sk_2}(x_2)] \leq negl(\lambda)$, where the probability takes over the randomly chosen of the inputs x_1, x_2.

4.2 The Construction

Now, we present the construction of the wPRF with efficient protocols from the LWR assumption. The constructed scheme F, which is described below, works with a domain $\mathbb{Z}_q^{m\times n}$, a range $\mathbb{Z}_p^m$, and a key space $\mathbb{Z}_q^n$, where n, m, p, q are positive integers that n is the security parameter, $p \geq 2$, $\gamma = q/p \in n^{\omega(1)}$ is an odd integer, and $m \geq n \cdot (\log q + 1)/(\log p - 1)$.

– **KeyGen.** The key generation algorithm samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and outputs $sk = \boldsymbol{s}$.
– **Eval.** On input an input $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, the evaluation algorithm outputs $\boldsymbol{y} = \lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rceil_p$.

Next, we present ZKAoK proving the correct evaluation of the function, i.e. given $\boldsymbol{s} \in \mathbb{Z}_q^n$, $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$ and $\boldsymbol{y} \in \mathbb{Z}_p^m$, proving that

$$\lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rceil_p = \boldsymbol{y} \mod p \tag{6}$$

Let $\boldsymbol{e} = \gamma \cdot \boldsymbol{y} - \boldsymbol{A} \cdot \boldsymbol{s}$, then we rewrite Equation (6) as follows.

$$\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} = \gamma \cdot \boldsymbol{y} \mod q \tag{7}$$

Note that if Equation (6) holds, then we have $\boldsymbol{e} \in [-\beta, \beta]^m$, where we define $\beta = \lfloor \gamma/2 \rfloor$ in this section. Also, if there exists $\boldsymbol{e} \in [-\beta, \beta]^m$ satisfying Equation (7), then we can also conclude that Equation (6) holds. Thus, proving Equation (6) is equal to proving that there exists $\boldsymbol{e} \in [-\beta, \beta]^m$ satisfying Equation (7). We give the concrete proof of statements for Equation (7) with different requirements (namely, with hidden $\boldsymbol{s}$, with hidden $\boldsymbol{s}, \boldsymbol{y}$, with hidden $\boldsymbol{s}, \boldsymbol{A}$, and with hidden $\boldsymbol{s}, \boldsymbol{A}, \boldsymbol{y}$) in Appendix D.

Then we move on proving the incorrect evaluation of the function, i.e. given $\boldsymbol{s} \in \mathbb{Z}_q^n$, $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$ and $\boldsymbol{y} \in \mathbb{Z}_p^m$, proving that

$$\lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rceil_p \neq \boldsymbol{y} \mod p \tag{8}$$

We can also transform Equation (8) as follows.

$$\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} \neq \gamma \cdot \boldsymbol{y} \mod q \tag{9}$$

However, proving that there exists $\boldsymbol{e} \in [-\beta, \beta]^m$ satisfying Equation (9) does not lead to Equation (8). This is because even Equation (8) does not hold (i.e. $\lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rceil_p = \boldsymbol{y} \mod p$), one can make Equation (9) to hold by using a fake $\boldsymbol{e}$. So, we must also prove the correctness of e, namely, the real number vector $\frac{\boldsymbol{e}}{\gamma}$ being the correct rounding error for $\frac{1}{\gamma} \cdot \boldsymbol{A} \cdot \boldsymbol{s}$. This can be done by proving that $\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$ equals to a vector with all elements being a multiple of γ, which implies that $\frac{1}{\gamma} \cdot \boldsymbol{A} \cdot \boldsymbol{s} + \frac{\boldsymbol{e}}{\gamma} = \boldsymbol{y}' \mod p$ for some vector in $\mathbb{Z}_p^m$, i.e. $\boldsymbol{e}$ is the correct rounding error for $\frac{1}{\gamma} \cdot \boldsymbol{A} \cdot \boldsymbol{s}$. In summary, we reduce the task of proving Equation (8) to the task of proving that there exists $\boldsymbol{e} \in [-\beta, \beta]^m$ and $\boldsymbol{y}' \in \mathbb{Z}_p^m$ satisfying Equation Equation (10) defined as follows.

$$\begin{cases} \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} - \gamma \cdot \boldsymbol{y}' = 0 \mod q \\ \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} \neq \gamma \cdot \boldsymbol{y} \mod q \end{cases} \tag{10}$$

This can be handled by the extended abstract Stern's protocol defined in Sec. 3, and we only need to provide abstract Stern's protocols for underlying abstract Stern's relations, which has already been provided in Appendix D. Note that, we should always use the protcol proving a relation with hidden output for the first equation.

Security of F is guaranteed by Theorem 4.1 stated as follows.

Theorem 4.1. *If the $\mathrm{LWR}_{n,q,p}$ assumption holds, and $m \geq n \cdot (\log q + 1)/(\log p - 1)$, then F is a secure wPRF.*

Proof. Weak pseudorandomness of F comes from the $\mathrm{LWR}_{n,q,p}$ assumption directly.

The uniqueness property requires that for a matrix $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, the probability that $\exists \boldsymbol{s_1}, \boldsymbol{s_2} \in \mathbb{Z}_q^n$ satisfying $\boldsymbol{s_1} \neq \boldsymbol{s_2} \mod q$ and $\lfloor \boldsymbol{A} \cdot \boldsymbol{s_1} \rceil_p = \lfloor \boldsymbol{A} \cdot \boldsymbol{s_2} \rceil_p \mod p$ is negligible. Note that the latter equation implies that $\exists \boldsymbol{e} \in (-\gamma, \gamma)^m$ that $\boldsymbol{e} = \boldsymbol{A} \cdot (\boldsymbol{s_1} - \boldsymbol{s_2}) \mod q$. So, it is sufficient to prove that the probability $\mathfrak{p}$ that $\exists \boldsymbol{d} \in \mathbb{Z}_q^n, \boldsymbol{e} \in (-\gamma, \gamma)^m$ satisfying $\boldsymbol{d} \neq \boldsymbol{0} \mod q$ and $\boldsymbol{A} \cdot \boldsymbol{d} = \boldsymbol{e} \mod q$ is negligible. Note that for each fixed $\boldsymbol{e}$, the probability that $\exists \boldsymbol{d} \neq \boldsymbol{0}$ that $\boldsymbol{e} = \boldsymbol{A} \cdot \boldsymbol{d}$ is less than $\frac{nq^n}{q^m}$. This is caculated in two cases: 1) if $\boldsymbol{e} = \boldsymbol{0}$, then it means that $\boldsymbol{A}$ has a rank less than n, which is of probability less than $\frac{nq^{n-1}}{q^m}$; 2) if $\boldsymbol{e} \neq \boldsymbol{0}$, then it means that $(\boldsymbol{e}, \boldsymbol{A})$ has a rank less than $n+1$, which is of probability less than $\frac{nq^n}{q^m}$ (note that $\boldsymbol{e}$ is a fixed non-zero vector). Finally, by the union bound, the probability $\mathfrak{p} \leq \frac{(2\gamma-1)^m \cdot n \cdot q^n}{q^m} \leq \frac{2^m \cdot n \cdot q^n}{p^m} \leq \frac{n \cdot q^n}{2^{n(\log q+1)}} = \frac{n}{2^n}$, which is negligible in n. That completes the proof. □

Moreover, if we further require that $m \geq 2n \cdot (\log q + 1)/(\log p - 1)$, then we can prove that the constructed wPRF with efficient protocol has strong uniqueness:

Proof. The strong uniqueness property requires that for two matrices $\boldsymbol{A}_1, \boldsymbol{A}_2 \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, the probability that $\exists \boldsymbol{s_1}, \boldsymbol{s_2} \in \mathbb{Z}_q^n$ satisfying $\boldsymbol{s_1} \neq \boldsymbol{s_2} \mod q$ and $\lfloor \boldsymbol{A_1} \cdot \boldsymbol{s_1} \rceil_p = \lfloor \boldsymbol{A_2} \cdot \boldsymbol{s_2} \rceil_p \mod p$ is negligible.[7]. Note that the latter equation implies that $\exists \boldsymbol{e} \in (-\gamma, \gamma)^m$ that $\boldsymbol{e} = \boldsymbol{A_1} \cdot \boldsymbol{s_1} - \boldsymbol{A_2} \cdot \boldsymbol{s_2} \mod q$. So, it is sufficient to prove that the probability $\mathfrak{p}$ that $\exists \boldsymbol{d}, \in \mathbb{Z}_q^{2n}, \boldsymbol{e} \in (-\gamma, \gamma)^m$ satisfying $\boldsymbol{d} \neq \boldsymbol{0} \mod q$ and $\boldsymbol{A} \cdot \boldsymbol{d} = \boldsymbol{e} \mod q$ is negligible, where $\boldsymbol{A} = (\boldsymbol{A}_1, \boldsymbol{A}_2)$. The remaining part is identical to that in the proof of Theorem 4.1, but as the random matrix $\boldsymbol{A}$ is uniform in $\mathbb{Z}_q^{m\times 2n}$ now, we need $m \geq 2n \cdot (\log q + 1)/(\log p - 1)$ to enable the inequalities. □

5 Applications

In this section, we demonstrate how to employ techniques and primitives presented in this work to construct concrete applications. In particular, we focus on the constructions of the linkable ring signature scheme, k-times anonymous authentication protocol, and blacklistable anonymous credential systems. Our construction follows the classical design principle of these schemes [ACST06, TFS04, TAKS07], and enjoys many features achieved by current constructions. Moreover, due to the primitives and techniques introduced in this work, our constructions are simple and clean in concept. Besides, the use of lattice also brings some new advantages comapred to previous constructions of these schemes.

In the remianing part of this secion, we only provide constructions of these schemes. The formal definition and security model of these schemes are provided in Appendix E, and the detailed security proof of the constructions are given in Appendix F.

[7] Here, we can prove an even stronger version, which does not reuqire $\boldsymbol{s_1} \neq \boldsymbol{s_2} \mod q$, but only require that either $\boldsymbol{s_1} \neq 0 \mod q$ or $\boldsymbol{s_2} \neq 0 \mod q$

5.1 Linkable Ring Signature

A linkable ring signature is a signature scheme that allows a user to sign on behalf of a spontaneous group of users, including himself, anonymously and allows anyone to link signatures signed by the same user. Here, we use the syntax and the security model in [ACST06], which are recalled in Appendix E.1. Our constructed linkable ring signature, which is built on the wPRF with efficient protocols $\mathsf{F} = (KeyGen', Eval')$ constructed in Sec. 4, the accumulator scheme $\mathsf{Acc} = (TSetup', TAcc', TWitness', TVerify')$ constructed in [LLNW16], and the supporting ZKAoKs, works as follows.

- **Setup.** Let n, m, p, q, k_1, k_2 be positive integers that n is the security parameter, $p \geq 2$, $\gamma = q/p \in n^{\omega(1)}$ is an odd integer, $m \geq n \cdot (\log q + 1)/(\log p - 1)$, $k_1 = \lceil \log p \rceil$, and $k_2 = \lceil \log q \rceil$. Let $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D_1}, \boldsymbol{D_2}) \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{n \times mk_1} \times \mathbb{Z}_q^{n \times 2nk_2}$, which can be sampled from some public random source, where $\boldsymbol{A}, \boldsymbol{B}$ are random inputs for the wPRF with efficient protocol, $\boldsymbol{D}_1$ is used to map an output of the wPRF with efficient protocol to the value space (values that can be accumulated by the accumulator) of the accumulator, and $\boldsymbol{D}_2$ is for the accumulator.
- **KeyGen.** On input a security parameter 1^n, the key generation algorithm runs $\boldsymbol{s} \leftarrow KeyGen'(1^n)$, computes $\boldsymbol{y} = Eval'(\boldsymbol{s}, \boldsymbol{A})$, and outputs $sk = \boldsymbol{s}$ and $pk = \boldsymbol{y}$.
- **Sign.** On input a secret key $\boldsymbol{s}$, a message $\mathfrak{m}$, and a set $\mathcal{R}$ of public keys that $\boldsymbol{y} = F_{\boldsymbol{s}}(\boldsymbol{A}) \in \mathcal{R}$, the signing algorithm first generates a set $\mathcal{R}'$ consisting of $\boldsymbol{y}'_{\boldsymbol{i}} = bin(\boldsymbol{D_1} \cdot bin(\boldsymbol{y_i}))$ for each element $\boldsymbol{y_i} \in \mathcal{R}$, and computes $\boldsymbol{y'} = bin(\boldsymbol{D_1} \cdot bin(\boldsymbol{y}))$. Then it computes $\boldsymbol{t} = F_{\boldsymbol{s}}(\boldsymbol{B})$, $\boldsymbol{u} = TAcc'_{\boldsymbol{D_2}}(\mathcal{R}')$, and $\mathfrak{w} = TWitness'_{\boldsymbol{D_2}}(\mathcal{R}', \boldsymbol{y'})$. Finally, it computes

$$\begin{aligned} \Pi = SPK\{(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{y'}, \mathfrak{w}) : \boldsymbol{y} = F_{\boldsymbol{s}}(\boldsymbol{A}) \wedge \boldsymbol{t} = F_{\boldsymbol{s}}(\boldsymbol{B}) \\ \wedge \boldsymbol{G_{n,q-1}} \cdot \boldsymbol{y'} = \boldsymbol{D_1} \cdot bin(\boldsymbol{y}) \mod q \wedge TVerify'_{\boldsymbol{D_2}}(\boldsymbol{u}, \boldsymbol{y'}, \mathfrak{w}) = 1\}[\mathfrak{m}] \end{aligned}$$

  , and outputs the signature $\Sigma = (\boldsymbol{t}, \Pi)$. Note that the term $\boldsymbol{y} = F_{\boldsymbol{s}}(\boldsymbol{A})$ and the term $\boldsymbol{t} = F_{\boldsymbol{s}}(\boldsymbol{B})$ can be proved under the abstract Stern's protocol as shown in Sec. 4; the term $TVerify'_{\boldsymbol{D_2}}(\boldsymbol{u}, \boldsymbol{y'}, \mathfrak{w}) = 1$ can also be proved under the abstract Stern's protocol as shown in [LLNW16]; and the term $\boldsymbol{G_{n,q-1}} \cdot \boldsymbol{y'} = \boldsymbol{D_1} \cdot bin(\boldsymbol{y}) \mod q$ can be proved by the abstract Stern's protocol [LLM+16a] directly. Thus, the whole proof can be generated by the extended abstract Stern's protcol proposed in Sec. 3.
- **Verify.** On input a message $\mathfrak{m}$, a set $\mathcal{R}$ of public keys, and a signature $\Sigma = (\boldsymbol{t}, \Pi)$, the verification algorithm first generates $\boldsymbol{u}$ as in the signing algorithm, then it checks the validity of the proof Π (with the help of $\boldsymbol{u}, \boldsymbol{t}$) and returns 1 iff Π is valid.
- **Link.** On input two signatures $\Sigma_1 = (\boldsymbol{t_1}, \Pi_1)$ and $\Sigma_2 = (\boldsymbol{t_2}, \Pi_2)$, the link algorithm outputs 1 iff $\boldsymbol{t_1} = \boldsymbol{t_2}$.

The Security. Security of the constructed linkable ring signature scheme is guaranteed by Theorem 5.1 stated as following, whose proof is put in Appendix F.1.

Theorem 5.1. *If* F *and* Acc *are secure wPRF with efficient protocols and secure accumulator scheme respectively, the underlying ZKAoK are secure, and the* $SIS^{\infty}_{n,mk_1,q,1}$ *assumption holds, then the linkable ring signature constructed here is a secure one.*

5.2 k-times Anonymous Authentication

A k-times anonymous authentication protocol allows users to authenticate themselves to an application provider in an anonymous and unlinkable manner after registering to a group manager, but up to k times. Here, we use the syntax and the security model in [TFS04], which are recalled in Appendix E.2. Our constructed k-times anonymous authentication protocol, which is built on the wPRF with efficient protocols (with strong uniqueness) $\mathsf{F} = (KeyGen', Eval')$ constructed in Sec. 4, the accumulator scheme $\mathsf{Acc} = (TSetup', TAcc', TWitness', TVerify')$ constructed in [LLNW16], the CL signature $\mathsf{CLS} = (KeyGen', Sign', Verify')$ constructed in [LLM+16a], a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times n}$, and the supporting ZKAoK, works as follows.

- **Public Parameters.** Let $n, m, p, q, k_1, k_2, \ell$ be positive integers that n is the security parameter, $p \geq 2$ is a prime, $\gamma = q/p \in n^{\omega(1)}$ is an odd integer, $m \geq 2n \cdot (\log q + 1)/(\log p - 1)$, $k_1 = \lceil \log p \rceil$, $k_2 = \lceil \log q \rceil$, and $\ell = \Theta(n)$. Let $(\boldsymbol{A}, \boldsymbol{E_1}, \boldsymbol{E_2}, \tilde{\boldsymbol{B}}, \boldsymbol{G_0}, \boldsymbol{G_1}, \boldsymbol{D_0}, \boldsymbol{D_1}) \xleftarrow{\$} \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{n\times 2mnk_2} \times \mathbb{Z}_q^{n\times 2nk_2} \times \mathbb{Z}_q^{n\times 2nk_2} \times \mathbb{Z}_q^{n\times \ell} \times \mathbb{Z}_q^{n\times 4nk_2} \times \mathbb{Z}_q^{2n\times 4nk_2} \times \mathbb{Z}_q^{2n\times 2mk_1}$, which can be sampled from some public random source, where $\boldsymbol{A}$ is random inputs for the wPRF with efficient protocols, $\boldsymbol{E}_1$ is used to map a pair of inputs of the wPRF with efficient protocols to the value space of the accumulator, $\boldsymbol{E}_2$ is for the accumulator, and $(\tilde{\boldsymbol{B}}, \boldsymbol{G_0}, \boldsymbol{G_1}, \boldsymbol{D_0}, \boldsymbol{D_1})$ are public parameters of the CL signature scheme. Here, we adapt the dimension of $\boldsymbol{D_1}$, to fit the length of the output of F, which will be signed in the registration phase. This will not affect the security of the CL signature scheme.
- **Setup.** In the setup procedure, the group manager generates the public key, secret key pair PK_{CL}, SK_{CL} of the CL signature scheme, and sets the group public key as PK_{CL} and the group secret key as SK_{CL}.
- **Join.** In this protocol, a user *uid* first generates his secret key $\boldsymbol{s} \leftarrow KeyGen'(1^n)$, then computes $\boldsymbol{y} = Eval'(\boldsymbol{s}, \boldsymbol{A})$. Then he sends $\boldsymbol{y}$ to the group manager. If $\boldsymbol{y}$ has not appeared in the identification list, the group manager adds $(uid, \boldsymbol{y})$ to the identification list and sign on $\boldsymbol{y}$ with SK_{CL}. It then sends the signature σ back to *uid*. Finally, the secret key of *uid* is $\boldsymbol{s}$ and his public key is $(\boldsymbol{y}, \sigma)$.
- **Bound Announcement.** In this protocol, an application provider AP publish (ID_{AP}, k_{AP}), where ID_{AP} is its identification and k_{AP} is the upper bound for each user to access its service. In this way, everyone in the system can compute the tag bases set $\mathcal{B}$ of the application provider, which consists of $(\boldsymbol{B_i}, \check{\boldsymbol{B}}_{\boldsymbol{i}}) = \mathcal{H}(ID_{AP}, k_{AP}, i)$ for $i \in [1, k]$.
- **Authentication.** In this protocol, a user *uid* with secret key $\boldsymbol{s}$ and public key $(\boldsymbol{y}, \sigma)$, who would like to access services of an application provider AP, first checks if he has already authenticated k_{AP} times to AP. If not, assuming that this is his ith time to access the services for AP, he then computes $(\boldsymbol{B}, \check{\boldsymbol{B}}) = \mathcal{H}(ID_{AP}, k_{AP}, i)$. He also generates a set $\mathcal{B}'$ consisting of $\boldsymbol{b'}_j = bin(\boldsymbol{E_1} \cdot bin(M2V(\boldsymbol{B}_j \| \check{\boldsymbol{B}}_j)))$ for each element $(\boldsymbol{B}_j, \check{\boldsymbol{B}}_j) \in \mathcal{B}$, and computes $\boldsymbol{b'} = bin(\boldsymbol{E_1} \cdot bin(M2V(\boldsymbol{B} \| \check{\boldsymbol{B}})))$. Then he computes $\boldsymbol{u} = TAcc'_{\boldsymbol{E_2}}(\mathcal{B}')$, and $\mathfrak{w} = TWitness'_{\boldsymbol{E_2}}(\mathcal{B}', \boldsymbol{b'})$. After that, he makes a request to AP. and gets a challenge $(c, \mathfrak{m}) \xleftarrow{\$} \mathbb{Z}_p \times \{0,1\}^n$ back. Then the user

computes $\boldsymbol{t} = F_s(\boldsymbol{B})$, $\check{\boldsymbol{t}} = F_s(\check{\boldsymbol{B}}) + c \cdot \boldsymbol{y} \mod p$, and

$$\begin{aligned} \Pi = SPK\{(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma) : \boldsymbol{y} = F_s(\boldsymbol{A}) \wedge Verify'(PK_{CL}, \boldsymbol{y}, \sigma) = 1 \\ \wedge \boldsymbol{t} = F_s(\boldsymbol{B}) \wedge \check{\boldsymbol{t}}' = F_s(\check{\boldsymbol{B}}) \wedge \gamma \cdot \check{\boldsymbol{t}}' + \gamma \cdot c \cdot \boldsymbol{y} = \gamma \check{\boldsymbol{t}} \mod q \\ \wedge \boldsymbol{G_{n,q-1}} \cdot \boldsymbol{b}' = \boldsymbol{E_1} \cdot bin(M2V(\boldsymbol{B} \| \check{\boldsymbol{B}})) \mod q \wedge Tverify'_{E_2}(\boldsymbol{u}, \boldsymbol{b}', \mathfrak{w}) = 1\}[\mathfrak{m}] \end{aligned}$$

He then sends $(\boldsymbol{t}, \check{\boldsymbol{t}}, \Pi)$ to AP. Note that the term $\boldsymbol{y} = F_s(\boldsymbol{A})$, the term $\boldsymbol{t} = F_s(\boldsymbol{B})$ and the term $\check{\boldsymbol{t}}' = F_s(\check{\boldsymbol{B}})$ can be proved under the abstract Stern's protocol as shown in Sec. 4; the term $Verify'(PK_{CL}, \boldsymbol{y}, \sigma) = 1$ can be proved under the abstract Stern's protocol as shown in [LLM+16a]; the term $TVerify'_{E_2}(\boldsymbol{u}, \boldsymbol{b}', \mathfrak{w}) = 1$ can also be proved under the abstract Stern's protocol as shown in [LLNW16]; and the term $\boldsymbol{G_{n,q-1}} \cdot \boldsymbol{b}' = \boldsymbol{E_1} \cdot bin(M2V(\boldsymbol{B} \| \check{\boldsymbol{B}})) \mod q$ and the term $\gamma \cdot \check{\boldsymbol{t}}' + \gamma \cdot c \cdot \boldsymbol{y} = \gamma \check{\boldsymbol{t}} \mod q$ can be proved by the abstract Stern's protocol [LLM+16a] directly. Thus, the whole proof can be generated by the extended abstract Stern's protcol proposed in Sec. 3. On receiving the response $(\boldsymbol{t}, \check{\boldsymbol{t}}, \Pi)$, AP first computes $\boldsymbol{u}$ in the same way as what the user *uid* did. Then it checks if Π is valid (with the help of $\boldsymbol{u}, \boldsymbol{t}, \check{\boldsymbol{t}}$) and stores $(\boldsymbol{t}, \check{\boldsymbol{t}}, c, \mathfrak{m}, \Pi)$ to its authentication log if this is the case. After that it checks if $\boldsymbol{t}$ has appeared before and accepts *uid* if $\boldsymbol{t}$ has not appeared and Π is valid.

– **PublicTracing.** On input an authentication log $\{(\boldsymbol{t_i}, \check{\boldsymbol{t}}_i, c_i, \mathfrak{m}_i, \Pi_i)\}_{i=1}^{L}$ of L elements and an identification list, the algorithm first initilize an empty set C. Then for each pair $i, j \in [1, L]$ that $\boldsymbol{t_i} = \boldsymbol{t_j}$, $c_i \neq c_j$ and both Π_i and Π_j are valid, the algorithm first computes $\boldsymbol{y} = (c_i - c_j)^{-1} \cdot (\check{\boldsymbol{t}}_i - \check{\boldsymbol{t}}_j)$. Note that since p is a prime, and $c_i \neq c_j$, $c_i - c_j$ is invertible. Then it searches $\boldsymbol{y}$ in the identification list and puts *uid* in C if it finds an item $(uid, \boldsymbol{y})$ in the identification list. Otherwise, it puts GM in C. Finally, the algorithm outputs the set C.

The Security. Security of the constructed k-times anonymous authentication protocol is guaranteed by Theorem 5.2 stated as following, whose proof is put in Appendix F.2.

Theorem 5.2. *If* F, ACC, *and* CLS *are secure wPRF with efficient protocols with stong uniqueness, secure accumulator scheme, and secure CL singature scheme respectively, the underlying ZKAoK are secure, the* $SIS^{\infty}_{n,mk_1,q,1}$ *assumption holds, and* $\mathcal{H}$ *is modeled as a random oracle, then the k-times anonymous authentication protocol constructed here is also secure.*

5.3 Blacklistable Anonymous Credentials

In a blacklistable anonymous credential system, users register themselves to a group manager, and then they can authenticate themselves to service providers in an anonymous and unlinkable manner, while those service providers can put misbehaved users into a blacklist to forbid them from accessing their services. Here, we use the syntax and the security model in [TAKS07], which are recalled in Appendix E.3. Our constructed blacklistable anonymous credential system, which is built on the wPRF with efficient protocols $\mathsf{F} = (KeyGen', Eval')$ constructed in Sec. 4, the CL signature $\mathsf{CLS} = (KeyGen', Sign', Verify')$ constructed in [LLM+16a], a cryptographic hash function $\mathcal{H} : \{0,1\}^n \to \mathbb{Z}_q^{m \times n}$, and the supporting ZKAoK, works as follows.

– **Public Parameters.** Let $n, m, p, q, k_1, k_2, \ell$ be positive integers that n is the security parameter, $p \geq 2$, $\gamma = q/p \in n^{\omega(1)}$ is an odd integer, $m \geq n \cdot (\log q + 1)/(\log p - 1)$, $k_1 = \lceil \log p \rceil$, $k_2 = \lceil \log q \rceil$, and $\ell = \Theta(n)$. Let $(\boldsymbol{A}, \tilde{\boldsymbol{B}}, \boldsymbol{G_0}, \boldsymbol{G_1}, \boldsymbol{D_0}, \boldsymbol{D_1}) \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{n \times 2nk_2} \times \mathbb{Z}_q^{n \times \ell} \times \mathbb{Z}_q^{n \times 4nk_2} \times \mathbb{Z}_q^{2n \times 4nk_2} \times \mathbb{Z}_q^{2n \times 2mk_1}$, which can be sampled from some public random source, where $\boldsymbol{A}$ is random inputs for the wPRF with efficient protocols, and $(\tilde{\boldsymbol{B}}, \boldsymbol{G_0}, \boldsymbol{G_1}, \boldsymbol{D_0}, \boldsymbol{D_1})$ are public parameters of the CL signature. Here, we adapt the dimension of $\boldsymbol{D_1}$, to fit the length of the output of F, which will be signed in the registration phase. This will not affect the security of the CL signature scheme.
– **Setup.** In the setup procedure, the group manager generates the public key, secret key pair PK_{CL}, SK_{CL} of the CL signature scheme, and sets the group public key as PK_{CL} and the group secret key as SK_{CL}.
– **Registration.** In this protocol, a user *uid* first generates his secret key $\boldsymbol{s} \leftarrow KeyGen'(1^n)$, then computes $\boldsymbol{y} = Eval'(\boldsymbol{s}, \boldsymbol{A})$. Then he sends $\boldsymbol{y}$ to the group manager. The group manager signs on $\boldsymbol{y}$ with SK_{CL}. It then sends the signature σ back to *uid*. Finally, the secret key of *uid* is $\boldsymbol{s}$ and his public key is $(\boldsymbol{y}, \sigma)$.
– **Authentication.** In this protocol, a user with secret key $\boldsymbol{s}$ and public key $(\boldsymbol{y}, \sigma)$ first make a request to the service provider, and gets a challenge $\mathfrak{m}$ as well as a blacklist $\mathcal{BL} = \{\mu_i, \boldsymbol{t_i}\}_{i \in [1, \|\mathcal{BL}\|]}$ back. Then for $i \in [1, \|\mathcal{BL}\|]$, the user computes $\boldsymbol{B_i} = \mathcal{H}(\mu_i)$ and checks if $t_i = F_s(\boldsymbol{B_i})$. He returns "failure" back to the service provider and aborts if such element in $\mathcal{BL}$ exists, and proceeds otherwise. Next, the user samples $\mu \xleftarrow{\$} \{0,1\}^n$, computes $\boldsymbol{B} = \mathcal{H}(\mu)$, $\boldsymbol{t} = F_s(\boldsymbol{B})$, and generates

$$\Pi = SPK\{(\boldsymbol{s}, \boldsymbol{y}, \sigma) : Verify'(PK_{CL}, \boldsymbol{y}, \sigma) = 1 \wedge \boldsymbol{y} = F_s(\boldsymbol{A}) \wedge \boldsymbol{t} = F_s(\boldsymbol{B}) \wedge (\bigwedge_{i=1}^{\|\mathcal{BL}\|} \boldsymbol{t_i} \neq F_s(\boldsymbol{B_i}))\}[\mathfrak{m}]$$

Then he returns $(\mu, \boldsymbol{t}, \Pi)$ to the service provider. Note that the term $\boldsymbol{y} = F_s(\boldsymbol{A})$, the term $\boldsymbol{t} = F_s(\boldsymbol{B})$ and the term $\boldsymbol{t_i} \neq F_s(\boldsymbol{B}_i)$ can be proved under the abstract Stern's protocol as shown in Sec. 4; and the term $Verify'(PK_{CL}, \boldsymbol{y}, \sigma) = 1$ can be proved under the abstract Stern's protocol as shown in [LLM+16a]. Thus, the whole proof can be generated by the extended abstract Stern's protcol proposed in Sec. 3.
On receiving the response $(\mu, \boldsymbol{t}, \Pi)$, the service provider accepts the user iff Π is valid. Note that the tuple $(\mu, \boldsymbol{t})$, which is the ticket for this authentication event, will be put in the blacklist of the service provider if the user misbehaves when accessing the services afterwards.

The Extensions. We remark that via slightly modifying the behavior of the user in the authentication protocol, we can upgrade the system to supporting fine-grained policies. For instance, to support a "d-strikes-out" policy, the user works identically to that in the basic case except that he returns "failure" only if its secret key is related to more than d items in the blacklist, and prove the following statement instead of the original one.

$$\Pi = SPK\{(\boldsymbol{s}, \boldsymbol{y}, \sigma) : \boldsymbol{y} = F_s(\boldsymbol{A}) \wedge Verify'(PK_{CL}, \boldsymbol{y}, \sigma) = 1 \wedge \boldsymbol{t} = F_s(\boldsymbol{B}) \wedge \|\{i \mid i \in [1, \|\mathcal{BL}\|], \boldsymbol{t_i} \neq F_s(\boldsymbol{B_i})\}\| \geq (\|\mathcal{BL}\| - d)\}[\mathfrak{m}]$$

Note that the term $\|\{i \mid i \in [1, \|\mathcal{BL}\|], \boldsymbol{t_i} \neq F_s(\boldsymbol{B_i})\}\| \geq (\|\mathcal{BL}\| - d)$ is in fact a threshold combination of underlying sub-statements (i.e. statements of of the form $\boldsymbol{t_i} \neq F_s(\boldsymbol{B_i})$), and can be expressed by a monotone span program. Thus, the user can complete via applying our extended abstract Stern's protocol.

The Security. Security of the constructed blacklistable anonymous credential system is guaranteed by Theorem 5.3 stated as following, whose proof is put in Appendix F.3.

Theorem 5.3. *If* F *and* CLS *are secure wPRF with efficient protocols and secure CL singature scheme respectively, the underlying ZKAoK are secure, and* $\mathcal{H}$ *is modeled as a random oracle, then the blacklistable anonymous credential system constructed here is also secure.*

References

[ACST06] Man Ho Au, Sherman SM Chow, Willy Susilo, and Patrick P Tsang. Short linkable ring signatures revisited. In *European Public Key Infrastructure Workshop*, pages 101–115. Springer, 2006.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, pages 99–108. ACM, 1996.

[AK12] Man Ho Au and Apu Kapadia. Perm: Practical reputation-based blacklisting without ttps. In *CCS*, pages 929–940. ACM, 2012.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited. In *CRYPTO*, pages 57–74. Springer, 2013.

[AKS12] Man Ho Au, Apu Kapadia, and Willy Susilo. Blacr: Ttp-free blacklistable anonymous credentials with reputation. In *NDSS*, 2012.

[ASA16] Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from lwe to lwr. *IACR Cryptology ePrint Archive*, 2016:589, 2016.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-taa. In *SCN*, pages 111–125. Springer, 2006.

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *CCS*, pages 132–145. ACM, 2004.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT*, pages 551–572. Springer, 2014.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable vrfs revisited. In *International Conference on Pairing-Based Cryptography*, pages 114–131. Springer, 2009.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, pages 201–218. Springer, 2010.

[BDM93] Josh Benaloh and Michael De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *EUROCRYPT*, pages 274–285. Springer, 1993.

[BDOP16] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris Peikert. Efficient commitments and zero-knowledge protocols from ring-sis with applications to lattice-based threshold cryptosystems. Cryptology ePrint Archive, Report 2016/997, 2016. `http://eprint.iacr.org/2016/997`.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *TCC*, pages 209–224. Springer, 2016.

[BGRV09] Zvika Brakerski, Shafi Goldwasser, Guy N Rothblum, and Vinod Vaikuntanathan. Weak verifiable random functions. In *TCC*, pages 558–576. Springer, 2009.

[BK10] Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. *IACR Cryptology ePrint Archive*, 2010:86, 2010.

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS*, pages 305–325. Springer, 2015.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737. Springer, 2012.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187. Springer, 1994.

[Cha81] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[Cha83] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *CCS*, pages 201–210. ACM, 2006.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118. Springer, 2001.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks*, pages 268–289. Springer, 2002.

[CL07] Melissa Chase and Anna Lysyanskaya. Simulatable vrfs with applications to multi-theorem nizk. In *CRYPTO*, pages 303–322. Springer, 2007.

[CNR12] Jan Camenisch, Gregory Neven, and Markus Rückert. Fully anonymous attribute tokens from lattices. In *SCN*, pages 57–75. Springer, 2012.

[CPS+16a] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved or-composition of sigma-protocols. In *TCC*, pages 112–141. Springer, 2016.

[CPS+16b] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline or composition of sigma protocols. In *EUROCRYPT*, pages 63–92. Springer, 2016.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, pages 126–144. Springer, 2003.

[CVH91] David Chaum and Eugène Van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

[DLA12] Ivan Damgård and Adriana López-Alt. Zero-knowledge proofs with low amortized communication from lattice assumptions. In *SCN*, pages 38–56. Springer, 2012.

[DSDCPY94] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of szk. In *FOCS*, pages 454–465. IEEE, 1994.

[ELL+15] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A provably secure group signature scheme from code-based assumptions. In *ASIACRYPT*, pages 260–285. Springer, 2015.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194. Springer, 1986.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *PKC*, pages 181–200. Springer, 2007.

[Fuj11] Eiichiro Fujisaki. Sub-linear size traceable ring signatures without random oracles. In *CT-RSA*, pages 393–415. Springer, 2011.

[GG98] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *STOC*, pages 1–9. ACM, 1998.

[GKV10] S Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT*, pages 395–412. Springer, 2010.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30. ACM, 2007.

[JKPT12] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *ASIACRYPT*, pages 663–680. Springer, 2012.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389. Springer, 2008.

[LLLS13] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *ASIACRYPT*, pages 41–61. Springer, 2013.

[LLM+16a] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT*, pages 373–403. Springer, 2016.

[LLM+16b] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *ASIACRYPT*, pages 101–131. Springer, 2016.

[LLNW14] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *PKC*, pages 345–361. Springer, 2014.

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT*, pages 1–31. Springer, 2016.

[LMN16] Benoît Libert, Fabrice Mouhartem, and Khoa Nguyen. A lattice-based group signature scheme with message-dependent opening. In *ACNS*, pages 137–155. Springer, 2016.

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *PKC*, pages 107–124. Springer, 2013.

[LNW15] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: simpler, tighter, shorter, ring-based. In *PKC*, pages 427–449. Springer, 2015.

[LNWX17] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity with ease. Cryptology ePrint Archive, Report 2017/353, 2017. `http://eprint.iacr.org/2017/353`.

[LW05] Joseph Liu and Duncan Wong. Linkable ring signatures: Security models and new schemes. *Computational Science and Its Applications–ICCSA 2005*, pages 88–89, 2005.

[LWW04] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *ACISP*, pages 325–335. Springer, 2004.

[Lyu08] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC*, pages 162–179. Springer, 2008.

[Lyu09] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616. Springer, 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755. Springer, 2012.

[MB00] Katelyn YA McKenna and John A Bargh. Plan 9 from cyberspace: The implications of the internet for personality and social psychology. *Personality and social psychology review*, 4(1):57–75, 2000.

[MRV99] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *FOCS*, pages 120–130. IEEE, 1999.

[MV03] Daniele Micciancio and Salil P Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *CRYPTO*, pages 282–298. Springer, 2003.

[NSN05] Lan Nguyen and Rei Safavi-Naini. Dynamic k-times anonymous authentication. In *ACNS*, pages 318–333. Springer, 2005.

[NZZ15] Phong Q Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In *PKC*, pages 401–426. Springer, 2015.

[P+16] Chris Peikert et al. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.

[PV08] Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *CRYPTO*, pages 536–553. Springer, 2008.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

[RST01] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565. Springer, 2001.

[Rüc10] Markus Rückert. Lattice-based blind signatures. In *ASIACRYPT*, pages 413–430. Springer, 2010.

[Ste93] Jacques Stern. A new identification scheme based on syndrome decoding. In *CRYPTO*, pages 13–21. Springer, 1993.

[TAKS07] Patrick P Tsang, Man Ho Au, Apu Kapadia, and Sean W Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *CCS*, pages 72–81. ACM, 2007.

[TAKS08] Patrick P Tsang, Man Ho Au, Apu Kapadia, and Sean W Smith. Perea: Towards practical ttp-free revocation in anonymous authentication. In *CCS*, pages 333–344. ACM, 2008.

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. K-times anonymous authentication. In *ASIACRYPT*, pages 308–322. Springer, 2004.

[TS06] Isamu Teranishi and Kazue Sako. K-times anonymous authentication with a constant proving cost. In *PKC*, pages 525–542. Springer, 2006.

[TW05] Patrick P Tsang and Victor K Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *International Conference on Information Security Practice and Experience*, pages 48–60. Springer, 2005.

[TWC+04] Patrick P Tsang, Victor K Wei, Tony K Chan, Man Ho Au, Joseph K Liu, and Duncan S Wong. Separable linkable threshold ring signatures. In *International Conference on Cryptology in India*, pages 384–398. Springer, 2004.

[wik] Wikiproject on open proxies/help:blocked. `https://meta.wikimedia.org/wiki/WikiProject_on_open_proxies/Help:blocked`. Accessed: 2017-05-12.

[XXW13] Xiang Xie, Rui Xue, and Minqian Wang. Zero knowledge proofs from ring-lwe. In *International Conference on Cryptology and Network Security*, pages 57–73. Springer, 2013.

[YAY17] Zuoxia Yu, Manho Au, and Rupeng Yang. Manuscript, in preparation. 2017.

[Zim69] Philip G Zimbardo. The human choice: Individuation, reason, and order versus deindividuation, impulse, and chaos. In *Nebraska symposium on motivation*. University of Nebraska press, 1969.

A Useful Tools for Handling Vectors and Defining Permutations

Here we review a few current tools and techniques used to decompose and extend vectors, and to define permutations. For the sake of compatibility, we will assign new symbols for these tools.

First, we recall techniques to decompose an m-dimension vector with infinity norm not exceeding B into a vector with infinity norm 1 for postive integers m, B. The simplest way to complete this task is to use the binary representation of each element of the vector. Formally, we write this procedure as $bin(\cdot) : [-B, B]^m \to [-1, 1]^{m\lceil \log B+1 \rceil}$. To reverse the decomposition one can use the matrix $\boldsymbol{G}_{m,B}$, which is defined as $\boldsymbol{G}_{m,B} = I_m \otimes (1, 2, \ldots, 2^{\lceil \log B+1 \rceil - 1})$. Obviously, we have $\boldsymbol{G}_{m,B} \cdot bin(\boldsymbol{v}) = \boldsymbol{v}$ for any $\boldsymbol{v} \in [-B, B]^m$. However, using the binary representation will lead to a gap between the norm of the witness vector and that of the extracted vector. To removes this gap, one can use the decomposition technique presented in [LNSW13]. Here, we define $\delta_B = \lfloor \log B \rfloor + 1$ and define a sequence $\boldsymbol{b}_B = (B_1, \ldots, B_{\delta_B})^\intercal$ that $B_j = \lfloor (B + 2^{j-1})/2^j \rfloor$ for $j \in [1, \delta_B]$. As stated in [LNSW13], for any number $u \in [-B, B]$, one can efficiently decompose u into a sequence $u_1, \ldots u_{\delta_B}$ that $u_j \in \{-1, 0, 1\}$ for $j \in [1, \delta_B]$ and $\sum_{j=1}^{\delta_B} B_j \cdot u_j = u$, while for any number $u \notin [-B, B]$, it is impossible to decompose u into sequences of $\{-1, 0, 1\}$ since the sequence $\boldsymbol{b}_B$ satisfies $\sum_{j=1}^{\delta_B} B_j = B$. So, to decompose a vector in $[-B, B]^m$, one just needs

to use the above method to decompose each element and combines the resulting sequences. Formally, we write this procedure as the function $Dec_{m,B}(\cdot) : [-B, B]^m \rightarrow [-1,1]^{m\delta_B}$. We remark that if the input vector is non-negative, then the output of the function $Dec_{m,B}(\cdot)$ is a binary vector. To reverse the decomposition, one can use the matrix $\boldsymbol{K}_{m,B} = \boldsymbol{I}_m \otimes \boldsymbol{b}_B^\intercal \in \mathbb{Z}^{m\times m\delta_B}$. It is easy to check that for any vector $\boldsymbol{v} \in [-B, B]^m$, we have $\boldsymbol{v} = \boldsymbol{K}_{m,B} \cdot Dec_{m,B}(\boldsymbol{v})$.

Then, we recall techniques to extend a n-dimension vector with infinity norm 1 to have constant numbers of -1,0,1 for a positive integer n. As shown in [LNSW13], this can be achieved by appending a vector in $\{-1,0,1\}^{2n}$ with exactly $(n - n_{-1})$ elements of value -1, $(n - n_0)$ elements of value 0, and $(n - n_1)$ elements of value 1 to the input vector, where n_{-1}, n_0, n_1 are numbers of elements of value $-1, 0, 1$ in the input vector respectively. Formally, we write this extension procedure as the function $Ext3_n(\cdot) : [-1,1]^n \rightarrow \mathcal{B}_n^3$, where we define $\mathcal{B}_n^3 \subseteq \{-1,0,1\}^{3n}$ as the set of vectors with exactly n elements of value -1, n elements of value 0, and n elements of value 1. For simplicity, we also define $DecExt3_{m,B}(\cdot) = Ext3_{m\delta_B}(Dec_{m,B}(\cdot))$ for the combined decomposition and extension procedures. Also, we define $\hat{\boldsymbol{K}}_{m,B,3} = (\boldsymbol{K}_{m,B} \| 0^{m\times 2m\delta_B}) \in \mathbb{Z}^{m\times 3m\delta_B}$, and it is easy to check that $\boldsymbol{v} = \hat{\boldsymbol{K}}_{m,B,3} \cdot DecExt3_{m,B}(\boldsymbol{v})$ for any vector $\boldsymbol{v} \in [-B, B]^m$. Sometimes, we need to extend binary vectors, which may be generated by decomposing non-negative vectors. In this case, we can extend the vector without adding additional -1s. Formally, we define this extension procedure as the function $Ext2_n(\cdot) : [-1,1]^n \rightarrow \mathcal{B}_n^2$, where we define $\mathcal{B}_n^2 \subseteq \{0,1\}^{2n}$ as the set of vectors with exactly n elements of value 0, and n elements of value 1. Likewise, we define $DecExt2_{m,B}(\cdot) = Ext2_{m\delta_B}(Dec_{m,B}(\cdot))$, with a restricted domain $[0, B]^m$. Also, we define $\hat{\boldsymbol{K}}_{m,B,2} = (\boldsymbol{K}_{m,B} \| 0^{m\times m\delta_B}) \in \mathbb{Z}^{m\times 2m\delta_B}$, and it is easy to check that $\boldsymbol{v} = \hat{\boldsymbol{K}}_{m,B,2} \cdot DecExt2_{m,B}(\boldsymbol{v})$ for any $\boldsymbol{v} \in [0, B]^m$. In [ELL+15], Ezerman et.al. also presented another approach to extend binary vectors, which can achieve fine-grained control over bits of the vectors. More precisely, for an n-dimension vector $\boldsymbol{v}$, the extended vector $\hat{\boldsymbol{v}} = (1 - \boldsymbol{v}[1], \boldsymbol{v}[1], 1 - \boldsymbol{v}[2], \boldsymbol{v}[2], \ldots, 1 - \boldsymbol{v}[n], \boldsymbol{v}[n])$. Formally, we define this extension procedure as the function $Enc2_n(\cdot) : \{0,1\}^n \rightarrow \mathcal{D}_n$, where we define $\mathcal{D}_n = \{(0,1) \cup (1,0)\}^n$. Also, we define $DecEnc2_{m,B}(\cdot) = Enc2_{m\delta_B}(Dec_{m,B}(\cdot))$, whose domain is $[0, B]^m$. In addition, we define $\hat{\boldsymbol{J}}_{m,B,2}$ as an $(m \times 2m\delta_B)$-dimension matrix, whose $2i - 1$th column is $\boldsymbol{0}$, and $2i$th column is the ith column of $\boldsymbol{K}_{m,B}$, for $i \in [1, m\delta_B]$. It is easy to check that $\boldsymbol{v} = \hat{\boldsymbol{J}}_{m,B,2} \cdot DecEnc2_{m,B}(\boldsymbol{v})$ for any $\boldsymbol{v} \in [0, B]^m$.

Besides, we will also employ the technique proposed in [LLM+16b] for handling quadratic relations of vectors. More precisely, given integers l_1, l_2, k_1, k_2 that $l_1/k_1 = l_2/k_2$, we define (in a slightly different way from that in [LLM+16b]) $Expand_{l_1,l_2,k_1,k_2}(\cdot, \cdot)$ as a function that takes as input two vectors $\boldsymbol{x} \in \mathcal{D}_{l_1}$ and $\boldsymbol{y} \in \mathcal{D}_{l_2}$, and outputs a vector $\boldsymbol{z} \in \mathbb{Z}^{4l_1k_2}$, that for $i \in [1, 4l_1k_2]$, $\boldsymbol{z}[i] = \boldsymbol{x}[i_1 \cdot 2k_1 + i_3 \cdot 2 + i_7 + 1] \cdot \boldsymbol{y}[i_1 \cdot 2k_2 + i_5 \cdot 2 + i_8 + 1]$, where $i_1 = \lfloor (i-1)/(4k_1k_2) \rfloor$, $i_2 = (i-1) \,\%\, (4k_1k_2)$, $i_3 = \lfloor i_2/(4k_2) \rfloor$, $i_4 = i_2 \,\%\, (4k_2)$, $i_5 = \lfloor i_4/4 \rfloor$, $i_6 = i_4 \,\%\, 4$, $i_7 = \lfloor i_6/2 \rfloor$, $i_8 = i_6 \,\%\, 2$. This technique is usually used to help prove that a vector is the product of a hidden matrix and a hidden vector. To complete this task, for positive integers m, n, β_1, β_2, we also define $\tilde{\boldsymbol{Q}}_{m,\beta_1,\beta_2} = \boldsymbol{I}_{\boldsymbol{m}} \otimes (\boldsymbol{b}_{\beta_1} \otimes \boldsymbol{b}_{\beta_2})^\intercal \in \mathbb{Z}^{m\times m\delta_{\beta_1}\delta_{\beta_2}}$; $\hat{\boldsymbol{Q}}_{m,\beta_1,\beta_2}$ as an $(m \times 4m\delta_{\beta_1}\delta_{\beta_2})$-dimension matrix, of which the $4i$th column is the ith column of $\tilde{\boldsymbol{Q}}_{m,\beta_1,\beta_2}$ for $i \in [1, m\delta_{\beta_1}\delta_{\beta_2}]$ and the other columns are $\boldsymbol{0}$; and $\boldsymbol{Q}_{m,n,\beta_1,\beta_2} = (\hat{\boldsymbol{Q}}_{m,\beta_1,\beta_2} \| \ldots \| \hat{\boldsymbol{Q}}_{m,\beta_1,\beta_2}) \in \mathbb{Z}^{m\times(4mn\delta_{\beta_1}\delta_{\beta_2})}$,

which are compromised of n matrices $\hat{\boldsymbol{Q}}_{m,\beta_1,\beta_2}$. It is not hard to check that for any $m \times n$-dimension matrix $\boldsymbol{A}$ that all elements in $\boldsymbol{A}$ is from $[0,\beta_1]$, and any n-dimension vector $\boldsymbol{x}$ that all elements in $\boldsymbol{x}$ is from $[0,\beta_2]$, we have $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{Q}_{m,n,\beta_1,\beta_2} \cdot z$ where $z = Expand_{mn\delta_{\beta_1},n\delta_{\beta_2},m\delta_{\beta_1},\delta_{\beta_2}}(\hat{\boldsymbol{a}},\hat{\boldsymbol{x}})$, $\hat{\boldsymbol{a}} = DecEnc2_{mn,\beta_1}(M2V(\boldsymbol{A}))$, $\hat{\boldsymbol{x}} = DecEnc2_{n,\beta_2}(\boldsymbol{x})$.

Finally, we recall a few useful notions [LNSW13, ELL+15, LLM+16b] to define valid permutations. First, for any postive integer k, we define $\mathcal{P}_k$ as the set of all permutations of k elements. Then we define the permutation fucntion $\mathtt{F}_{\cdot}(\cdot)$, that for any postive integer k, any binary string $\boldsymbol{b} \in \{0,1\}^k$, and any $2k$-dimension vector $\boldsymbol{x}$, $\mathtt{F}_{\boldsymbol{b}}(\boldsymbol{x}) = (\boldsymbol{x}[\boldsymbol{b}[1]+1], \boldsymbol{x}[1-\boldsymbol{b}[1]+1], \ldots, \boldsymbol{x}[2(i-1)+\boldsymbol{b}[i]+1], \boldsymbol{x}[2(i-1)+1-\boldsymbol{b}[i]+1], \ldots, \boldsymbol{x}[2n-2+\boldsymbol{b}[n]+1], \boldsymbol{x}[2n-2+1-\boldsymbol{b}[n]+1])^\intercal$. Besides, we define the permutation function $\mathtt{P}_{\cdot,\cdot,\cdot,\cdot}(\cdot)$, that for any positive integers l_1, l_2, k_1, k_2, any binary strings $\boldsymbol{b} \in \{0,1\}^{l_1}, \boldsymbol{d} \in \{0,1\}^{l_2}$, and any $4l_1k_2$-dimension vector z, $\mathtt{P}_{\boldsymbol{b},\boldsymbol{d},k_1,k_2}(z) = z'$, and for $i \in [1, 4l_1k_2]$ $z'[i] = z[i_1 \cdot 4k_1k_2 + i_3 \cdot 4k_2 + i_5 \cdot 4 + (i_7 \oplus \boldsymbol{b}[i_1 \cdot k_1 + i_3 + 1]) \cdot 2 + (i_8 \oplus \boldsymbol{d}[i_1 \cdot k_2 + i_5 + 1]) + 1]$, where $i_1 = \lfloor (i-1)/(4k_1k_2) \rfloor$, $i_2 = (i-1) \,\%\, (4k_1k_2)$, $i_3 = \lfloor i_2/(4k_2) \rfloor$, $i_4 = i_2 \,\%\, (4k_2)$, $i_5 = \lfloor i_4/4 \rfloor$, $i_6 = i_4 \,\%\, 4$, $i_7 = \lfloor i_6/2 \rfloor$, $i_8 = i_6 \,\%\, 2$. It is not hard to check that, for any positive integers l_1, l_2, k_1, k_2, any binary strings $\boldsymbol{b} \in \{0,1\}^{l_1}$, $\boldsymbol{d} \in \{0,1\}^{l_2}$, and any vectors $\boldsymbol{x} \in \mathcal{D}_{l_1}$, $\boldsymbol{y} \in \mathcal{D}_{l_2}$, let $z' = \mathtt{P}_{\boldsymbol{b},\boldsymbol{d},k_1,k_2}(z)$, $\boldsymbol{x}' = \mathtt{F}_{\boldsymbol{b}}(\boldsymbol{x})$, and $\boldsymbol{y}' = \mathtt{F}_{\boldsymbol{d}}(\boldsymbol{y})$, then we have

$$z = Expand_{l_1,l_2,k_1,k_2}(\boldsymbol{x},\boldsymbol{y}) \iff z' = Expand_{l_1,l_2,k_1,k_2}(\boldsymbol{x}',\boldsymbol{y}')$$

B Related Works

Privacy-Preserving Authentication with Accountable Anonymity. Linkable ring signature was presented by Liu et.al. in [LWW04]. A linkable ring signature scheme allows a user to anonymously sign on behalf of a spontaneous group of users, including himself, and allows anyone to link signatures signed by the same user. Thus, it is suitable for scenarios where "authenticating more than one time" is regarded as misbehaviors, and is used to construct applications such as E-Voting [Cha81], E-Cash [Cha83] and Direct Anonymous Attestation [BCC04]. Then, a series of works [TWC+04,TW05,LW05,ACST06] following this line of research have been done to improve either the efficiency or the security of the scheme. Moreover, traceable ring signature, an improved version of linkable ring signature providing traceability, is presented [FS07] and constructed [FS07,Fuj11]. The K-times anonymous authentication protocol [TFS04,TS06] is also designed to restrict the number of times users are permitted to use applications. In particular, it allows the service providers to set the upper bound of the number of times a user could authenticate, and allows anyone to trace the identification of misbehaved users, thus provides a greater flexibility and a better security guarantee. Following works consider introducing new features into the protocol. More precisely, in [NSN05, ASM06], K-times anonymous authentication protocol allowing service providers to independently grant or revoke users are constructed, and in [CHK+06], protocols that allow users to authenticate k times in every time period are constructed. Both linkable ring signature and K-times anonymous authentication protocol can only protect security against misbehaviors led by authenticating too many times (e.g. double spending). To overcome

this restriction, in [TAKS07, TAKS08], blacklistable anonymous credential system is defined and constructed. It allows service providers to define misbehaviors via subjective judging and block users with these misbehaviors, thus can be applied to more general scenarios. Later, in [AKS12, AK12], blacklistable anonymous credential systems supporting "reward" for rewarding behaviors and fine-grained policy are also proposed to provide a better functionality for authentication.

Lattice-Based Privacy-Preserving Authentication. There already exist several works constructing privacy-preserving protocols in the lattice-based setting. Among them, the most actively researched one is the group signature scheme [CVH91]. Early lattice-based group signature schemes [GKV10, CNR12] have a signature size linear in the number of group users. Subsequent works [LLLS13, LLNW14, LNW15,NZZ15,LLNW16,LMN16,LLM+16a,LNWX17] reduce the signature size to be only logarithmic in the number of group users by employing more elaborate zero-knowledge proof techniques. In addition, the scheme constructed in [LLNW16] does not employ a lattice trapdoor, and thus admits a simpler construction and a better efficiency in practice. Moreover, dynamic group signature schemes are also constructed [LLNW14, LLM+16a, LNWX17], which can support either revocation of users or dynamic user enrollments or both. Compared to the group signature, there are fewer constructions of the ring signature scheme [RST01] in the lattice-based setting. Most constructions are based on the framework in [BK10], which will lead to a signature size linear in the number of ring members. In [LLNW16], via exploiting the accumulator in their construction, lattice-based ring signature schemes with signature size logarithmic in the number of ring members is also given. Besides, there are also some other lattice-based privacy-preserving authentication schemes constructed, e.g. blind signature [Rüc10], group encryption [LLM+16b], anonymous credential [LLM+16a], etc.

Most of these applications are built on lattice-based zero-knowledge proof (argument) of knowledge, which was initialized by Goldreich and Goldwasser in [GG98]. Early works [GG98, MV03, PV08] consider proofs for worst-case lattice-based problems, and are hardly applicable in practice. Later, in [LLLS13], zero-knowledge proof of knowledge for the ISIS problem, which is commonly used in practical constructions, is constructed by modifying the identification scheme in [Lyu08, Lyu09, Lyu12]. There are several works following this line of research [NZZ15, BCK+14, BKLP15, BDOP16]. Especially, the proof system [BKLP15] constructed in the ideal lattice-based setting is the most efficient zero-knowledge proofs for lattice-related languages. However, proof systems constructed following this paradigm suffer from the problem of extraction gap, namely the extracted witness may be much larger than the used witness, thus are not suitable for many application scenarios. To solve this problem, in [LNSW13], zero-knowledge arguments of knowledge without extraction gap are proposed via modifying the Stern's identification scheme [Ste93]. Aiming at supporting more advanced relations, extensive works have been done [XXW13, LLNW14, LNW15, ELL+15, LLNW16, LLM+16a, LLM+16b]. In particular, in [LLM+16a], a general framework for proving a large class of relations satisfying some specific structure is presented. Besides, one can also construct lattice-based zero-knowledge proofs from a proper multi-party computation protocols [BD10, DLA12] by employing the IKOS transform

[IKOS07]. Apart from zero-knowledge proofs, some other underlying primitives for constructing privacy-preserving authentication protocols are also constructed in the lattice-based setting recently, e.g. accumulator scheme [LLNW16] and CL signature [LLM+16a].

C Omitted Protocols in Sec. 3

C.1 An Alternative Protocol Proving $x \neq 0$

In this section, we present a new method for proving $\boldsymbol{x} \neq 0$, which is more flexiable but slightly less efficient compared to that in [LNSW13]. It can be applied to scenarios where the technique in [LNSW13] is not applicable due to its special requirement on the extension manner. As an imediate example, we will use the method presented here in Appendix C.2, where one has to introduce some tricks and reduce the efficiency of the whole protocol if they want to use the technique in [LNSW13].

Our method is based on the observation that a binary vector is not equal to $\mathbf{0}$ iff its 1 norm is not 0. In this way, to prove that a n-dimension vector $\boldsymbol{x} \neq \mathbf{0}$, the prover proves that there exists a non-negative number r less than n that $\|\boldsymbol{x}\|_1 - r = 1$. As the proof will be conducted over a module q, we also require that $q > n$. This condition can be satisfied in most cases in the lattice-based setting, but will not be satisfied when the modular is small, e.g. when the system builds on the LPN assumption. Nonetheless, our method is more flexiable than that in [LNSW13] since our method works well with any type of extension manner.

To demonstrate our method, we apply it on a small example. Let D', L', q be positive integers, let $\mathrm{VALID}' \subseteq \{0,1\}^{L'}$ be a set, and let $\mathcal{R}' = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D' \times L'} \times \mathbb{Z}_q^{D'} \times \mathrm{VALID}' : \boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q\}$ be an abstract Stern's relation. In fact, for a vector $\boldsymbol{x}' \in \mathrm{VALID}'$, it is always required to prove that bits of $\boldsymbol{x}'$ on some particular positions are not all 0, since $\boldsymbol{x}'$ is usually generated by extending the target vector (that needs to be proved non-zero) with additional 0s and 1s. Let $\boldsymbol{S} \in \mathbb{Z}_q^{n \times L'}$ be the matrix that can select the target vector from $\boldsymbol{x}'$. Then, given common input $\boldsymbol{P}' \in \mathbb{Z}_q^{D' \times L'}$, and prover's secret input $\boldsymbol{x}' \in \mathrm{VALID}'$, the prover needs to prove that

$$\begin{cases} \boldsymbol{P}' \cdot \boldsymbol{x}' = \mathbf{0} \mod q \\ \boldsymbol{S} \cdot \boldsymbol{x}' \neq \mathbf{0} \mod q \end{cases} \tag{11}$$

By using our method, we define $\boldsymbol{E} = (1, 1, \ldots, 1) \in \mathbb{Z}_q^{1 \times n}$, $r = \|\boldsymbol{x}'\|_1 - 1$ and transform Equation (11) as follows.

$$\begin{cases} \boldsymbol{P}' \cdot \boldsymbol{x}' = \mathbf{0} \mod q \\ \boldsymbol{E} \cdot \boldsymbol{S} \cdot \boldsymbol{x}' - r = 1 \mod q \end{cases} \tag{12}$$

Then, we decompose and extend Equation (12) and define the following matrices and vectors.

$$\begin{cases} \hat{\boldsymbol{r}} = DecEnc2_{1,n-1}(r), \boldsymbol{x} = (\boldsymbol{x'}^{\intercal} || \hat{\boldsymbol{r}}^{\intercal})^{\intercal} \\ \\ \boldsymbol{P} = \begin{pmatrix} \boldsymbol{P'} & 0^{D' \times 2\delta_{n-1}} \\ \boldsymbol{E} \cdot \boldsymbol{S} & -\hat{\boldsymbol{J}}_{1,n-1,2} \end{pmatrix} \\ \\ \boldsymbol{v} = ((0^{D'})^{\intercal} || 1)^{\intercal} \end{cases}$$

Then, we can transform Equation (12) as follows,

$$\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q$$

and define VALID= VALID$' \times \mathcal{D}_{\delta_{n-1}}$ that is a set of vectors of length $L = L' + 2\delta_{n-1}$. Obviously, $\boldsymbol{x} \in$VALID. It can be easily verified that this statement is equivalent to the original statement defined by Equation (11).

Then we need to specify the permutation used in the protocol. As the relation $\mathcal{R}'$ can be proved under the abstract Stern's protocol, there exists $\mathcal{S}', \mathtt{T}'$ that are valid for $\mathcal{R}'$. Then we define $\mathcal{S} = \mathcal{S}' \times \{0,1\}^{\delta_{n-1}}$. For $\pi = (\pi_1, \boldsymbol{b}_2) \in \mathcal{S}$, where $\pi_1 \in \mathcal{S}'$, and $\boldsymbol{b}_2 \in \{0,1\}^{\delta_{n-1}}$, and $\boldsymbol{w} = (\boldsymbol{w}_1^{\intercal} || \boldsymbol{w}_2^{\intercal})^{\intercal}$ where $\boldsymbol{w}_1$ is of length L', and $\boldsymbol{w}_2$ is of length $2\delta_{n-1}$ we define $\mathtt{T}_\pi(\boldsymbol{w}) = (\mathtt{T}'_{\pi_1}(\boldsymbol{w}_1)^{\intercal}, \mathtt{F}_{\boldsymbol{b}_2}(\boldsymbol{w}_2)^{\intercal})^{\intercal}$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

C.2 Protocols for The Monotone Span Program Composition of Abstract Stern's Relations and Polynomial Equation of Witnesses

In this section, we boost the protocol presented in Sec. 3.3 to a protocol for the relation combined by abstract Stern's relations via a monotone span program and a multivariate polynomial, i.e. the protocol proves that the indices of true statements in a set of statements, each of which is about an abstract Stern's relation, satisfy a monotone span program, and witnesses of these statements satisfy a multivariate polynomial equation.

Here, we use a vector $\boldsymbol{c} \in \mathbb{Z}^{m+1}$ and a matrix $\boldsymbol{E} \in \{0,1\}^{m \times n}$ to denote a multivariate polynomial with $m+1$ terms (including a constant term) on n variables and define the polynomial $\mathfrak{F}_{\boldsymbol{c},\boldsymbol{E}}(x_1, \ldots, x_n) = \sum_{i=1}^{m} \boldsymbol{c}[i] \cdot (\prod_{j=1}^{n} x_j^{\boldsymbol{E}[i][j]}) + \boldsymbol{c}[m+1]$. For the simplicity of description, we restrict the domain of the polynomial to $\{0,1\}^n$, and as a result, we only need to consider exponent of $0, 1$. However, this will not harm the usefulness of our framework since the final witness of an abstract Stern's protocol will be a vector of $\{-1, 0, 1\}$, and in principle it is not hard to extend the protocol presented in this section to protocols that can handle polynomials that in additional allows -1 appearing in its input.

More precisely, let t, q, k be positive integers, q' be a positive prime and $\tilde{q} = \mathrm{lcm}(q, q')$; let n be positive integers that $n < \tilde{q}$ and $\delta_n + 1 < \tilde{q}$; let $\gamma = \tilde{q}/q$ and $\gamma' = \tilde{q}/q'$; let $\mathcal{M} = (\mathbb{Z}_{q'}, \boldsymbol{M}, \rho)$ be a monotone span program accepting an access structure $\mathcal{A} \subseteq 2^{\{1,\ldots,t\}}$, where $\boldsymbol{M} \in \mathbb{Z}_{q'}^{\ell_1 \times \ell_2}$; for $i \in [1,t]$, let D_i, L_i be positive integers, let

$\mathrm{VALID}_i \subseteq \{-1, 0, 1\}^{L_i}$ be a set, and let $\mathcal{R}_i = \{((\boldsymbol{P}, \boldsymbol{v}), \boldsymbol{x}) \in \mathbb{Z}_q^{D_i \times L_i} \times \mathbb{Z}_q^{D_i} \times \mathrm{VALID}_i : \boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q\}$ be an abstract Stern's relation; let $\hat{D} = \sum_{i=1}^{t} D_i$, $\hat{L} = \sum_{i=1}^{t} L_i$; let $\mathcal{F} = \{\boldsymbol{c}_1, \boldsymbol{E}_1, \ldots, \boldsymbol{c}_k, \boldsymbol{E}_k\}$ be a set of k polynomials, that $\forall i \in [k]$, $\boldsymbol{c}_i \in \mathbb{Z}_q^{m_i+1}$ and $\boldsymbol{E}_i \in \{0, 1\}^{m_i \times n}$ for positive integers m_i; let $\boldsymbol{S} \in \{0, 1\}^{n \times \hat{L}}$ that has exactly one '1' in each row and are used to select suitable input to polynomials in $\mathcal{F}$, i.e. given any vector $\boldsymbol{x} \in \mathrm{VALID}_1 \times \ldots \times \mathrm{VALID}_t$, $\boldsymbol{S} \cdot \boldsymbol{x} \in \{0, 1\}^n$. Then, given common input $\boldsymbol{P}_i \in \mathbb{Z}_q^{D_i \times L_i}$, $\boldsymbol{v}_i \in \mathbb{Z}_q^{D_i}$, and prover's secret input $\boldsymbol{x}_i \in \mathrm{VALID}_i$ for $i \in [1, t]$, the prover needs to prove that

$$\begin{cases} \mathcal{T} = \{i \mid i \in [1, t] \wedge ((\boldsymbol{P}_i, \boldsymbol{v}_i), \boldsymbol{x}_i) \in \mathcal{R}_i\} \\ \mathcal{T} \in \mathcal{A} \\ \forall j \in [1, k],\ \mathfrak{F}_{\boldsymbol{c}_j, \boldsymbol{E}_j}(\boldsymbol{S} \cdot (\boldsymbol{x}_1^\intercal, \ldots, \boldsymbol{x}_t^\intercal)^\intercal) = 0 \mod q \end{cases} \tag{13}$$

Now, let b_i be a bit that indicates whether the ith statement is true for $i \in [1, t]$, and let $\boldsymbol{g} = (g_1, \ldots, g_{\ell_1})^\intercal \in \mathbb{Z}_{q'}^{\ell_1}$ be a vector that $\boldsymbol{g}^\intercal \cdot \boldsymbol{M} = (1, 0, \ldots, 0) \mod q'$ and $g_i = 0$ if $b_{\rho(i)} = 0$ for $i \in [1, \ell_1]$. Also, for $i \in [1, k]$, $j \in [1, m_i]$, let $y_{i,j} = \prod_{\iota=1}^{n} \boldsymbol{x'}[\iota]^{\boldsymbol{E}_i[j][\iota]}$, where we define $\boldsymbol{x'} = \boldsymbol{S} \cdot (\boldsymbol{x}_1^\intercal, \ldots, \boldsymbol{x}_t^\intercal)^\intercal$, and let $\boldsymbol{y} = (y_{1,1}, \ldots, y_{1,m_1}, \ldots, y_{k,1}, \ldots, y_{k,m_k})^\intercal$. Then we rewrite Equation (13) as follows.

$$\begin{pmatrix} \gamma \cdot \hat{\boldsymbol{P}} & -\gamma \cdot \hat{\boldsymbol{V}} & & \\ & & \gamma \cdot \boldsymbol{B} & \\ & & & \gamma' \cdot \boldsymbol{M}^\intercal \end{pmatrix} \cdot \begin{pmatrix} b_1 \cdot \boldsymbol{x}_1 \\ \vdots \\ b_t \cdot \boldsymbol{x}_t \\ b_1 \\ \vdots \\ b_t \\ \boldsymbol{y} \\ \boldsymbol{g} \end{pmatrix} = \begin{pmatrix} 0^{\hat{D}} \\ \gamma \cdot \boldsymbol{c} \\ \gamma' \cdot \boldsymbol{e} \end{pmatrix} \mod \tilde{q} \tag{14}$$

where in this section, we define $\hat{m} = \sum_{i=1}^{k} m_i$, $\boldsymbol{e} = (1, 0, \ldots, 0)^\intercal \in \{0, 1\}^{\ell_2}$, $\boldsymbol{c} = (-\boldsymbol{c}_1[m_1 + 1], \ldots, -\boldsymbol{c}_k[m_k + 1])^\intercal$ and define

$$\hat{\boldsymbol{P}} = \begin{pmatrix} \boldsymbol{P}_1 & & \\ & \ddots & \\ & & \boldsymbol{P}_t \end{pmatrix} \qquad \hat{\boldsymbol{V}} = \begin{pmatrix} \boldsymbol{v}_1 & & \\ & \ddots & \\ & & \boldsymbol{v}_t \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{c}_1[1] \ldots \boldsymbol{c}_1[m_1] & & \\ & \ddots & \\ & & \boldsymbol{c}_k[1] \ldots \boldsymbol{c}_k[m_k] \end{pmatrix}$$

Next, we need to decompose and extend Equation (14). The main difficulty is to argue that each $y_{i,j}$ is properly generated, namely equal to the product of some bits in the witness. One method to prove such non-linear equations is to extend the technique in [LLM+16b], which can prove that a bit is the product of two bits, to the higher dgeree cases. However, extending this method straightforwardly will introduce a communication cost that is exponential in the number of factors. Here, we present a new method to complete this task, which merely introduce a communication cost that is logarithmic in the number of factors.

To better describe our method, we abstract the task into proving that a bit z is the product of κ bits, i.e. $z = z_1 \cdot \ldots \cdot z_\kappa$ for bits $z_1, \ldots, z_\kappa$. This is equal to prove that $z = 1$

iff $z_i = 1$ for $i \in [1, \kappa]$, and we can further transform the statement into proving that $z = 1$ iff $\sum_{i=1}^{\kappa} z_i = \kappa$. Let $\iota = \kappa - \sum_{i=1}^{\kappa} z_i$, then it is sufficient to prove that (1) if $\iota = 0$ then $z = 1$ and (2) if $\iota \neq 0$ then $z = 0$. To prove the first part, we reduce it to proving $2\iota + z \neq 0$, and use the technique presented in Sec. C.1 to handle this inequation; and to prove the second part, we prove that there exists positive number u in $[0, \kappa]$ that $\iota - \bar{z} \cdot u = 0$, which can also be proved under Stern's protocol; we remark that the proof of neither part will affect the proof of the other part, thus we can prove both parts simultaneously, i.e. completing the task, by combining these two arguments.

More formally, to transform Equation (14), we use $n_{i,j}$ to denote the the number of 1s in the jth row of the matrix $\boldsymbol{E}_i$ for $i \in [1, k]$, $j \in [1, m_i]$, define $\dot{n} = \sum_{i=1}^{k} \sum_{j=1}^{m_i} \delta_{n_{i,j}}$, $\ddot{n} = \sum_{i=1}^{k} \sum_{j=1}^{m_i} \delta_{\delta_{n_{i,j}}}$, $\dot{\ell} = 2\ell_1 \delta_{q'-1}$, define $\boldsymbol{h}_\kappa = (0, 1, 0, 1, \ldots, 0, 1)^\intercal \in \{0, 1\}^{2\kappa}$ and $\hat{\boldsymbol{I}}_\kappa = \boldsymbol{I}_\kappa \otimes \boldsymbol{h}_1^\intercal$ for any positive integer κ, and define matricies and vectors as follows.

$$
\begin{cases}
\acute{\boldsymbol{x}} = (b_1 \cdot \boldsymbol{x}_1^\intercal \| \ldots \| b_t \cdot \boldsymbol{x}_t^\intercal)^\intercal, \grave{\boldsymbol{x}} = (\bar{b}_1 \cdot \boldsymbol{x}_1^\intercal \| \ldots \| \bar{b}_t \cdot \boldsymbol{x}_t^\intercal)^\intercal \\
\acute{\boldsymbol{b}} = (b_1, \ldots, b_t)^\intercal, \grave{\boldsymbol{b}} = (\bar{b}_1, \ldots, \bar{b}_t)^\intercal, \hat{\boldsymbol{y}} = Enc2_{\hat{m}}(\boldsymbol{y}) \\
\forall i \in [1,k], j \in [1, m_i], d_{i,j} = n_{i,j} - \sum_{\iota=1}^{n} \boldsymbol{E}_i[j][\iota] \cdot \boldsymbol{x}'[\iota], \hat{\boldsymbol{d}}_{i,j} = DecEnc2_{1,n_{i,j}}(d_{i,j}) \\
r'_{i,j} = \boldsymbol{h}_{\delta_{n_{i,j}}}^\intercal \cdot \hat{\boldsymbol{d}}_{i,j}, \hat{\boldsymbol{r}'}_{i,j} = DecEnc2_{1,\delta_{n_{i,j}}}(r'_{i,j}) \\
r_{i,j} = r'_{i,j} + y_{i,j} - 1, \hat{\boldsymbol{r}}_{i,j} = DecEnc2_{1,\delta_{n_{i,j}}}(r_{i,j}) \\
\hat{\boldsymbol{d}} = (\hat{\boldsymbol{d}}_{1,1}^\intercal, \ldots, \hat{\boldsymbol{d}}_{k,m_k}^\intercal)^\intercal, \hat{\boldsymbol{r}} = (\hat{\boldsymbol{r}}_{1,1}^\intercal, \ldots, \hat{\boldsymbol{r}}_{k,m_k}^\intercal)^\intercal \\
\acute{\boldsymbol{r}}' = (\bar{y}_{1,1} \cdot \hat{\boldsymbol{r}}'^\intercal_{1,1}, \ldots, \bar{y}_{k,m_k} \cdot \hat{\boldsymbol{r}}'^\intercal_{k,m_k})^\intercal, \grave{\boldsymbol{r}}' = (y_{1,1} \cdot \hat{\boldsymbol{r}}'^\intercal_{1,1}, \ldots, y_{k,m_k} \cdot \hat{\boldsymbol{r}}'^\intercal_{k,m_k})^\intercal \\
\forall i \in [1, \ell_1], \hat{\boldsymbol{g}}_i = DecEnc2_{1,q'-1}(\boldsymbol{g}[i]) \\
\acute{\boldsymbol{g}} = (b_{\rho(1)} \cdot \hat{\boldsymbol{g}}_1^\intercal \| \ldots \| b_{\rho(\ell_1)} \cdot \hat{\boldsymbol{g}}_{\ell_1}^\intercal)^\intercal, \grave{\boldsymbol{g}} = (\bar{b}_{\rho(1)} \cdot \hat{\boldsymbol{g}}_1^\intercal \| \ldots \| \bar{b}_{\rho(\ell_1)} \cdot \hat{\boldsymbol{g}}_{\ell_1}^\intercal)^\intercal \\
\boldsymbol{x} = (\acute{\boldsymbol{x}}^\intercal \| \grave{\boldsymbol{x}}^\intercal \| \acute{\boldsymbol{b}}^\intercal \| \grave{\boldsymbol{b}}^\intercal \| \acute{\boldsymbol{g}}^\intercal \| \grave{\boldsymbol{g}}^\intercal \| \hat{\boldsymbol{y}} \| \hat{\boldsymbol{d}} \| \hat{\boldsymbol{r}} \| \acute{\boldsymbol{r}}' \| \grave{\boldsymbol{r}}')^\intercal \\
\\
\boldsymbol{E} = \begin{pmatrix} \boldsymbol{E}_1 \\ \vdots \\ \boldsymbol{E}_k \end{pmatrix}, \boldsymbol{L}' = \begin{pmatrix} \hat{\boldsymbol{J}}_{1,\delta_{n_{1,1}},2} & & \\ & \ddots & \\ & & \hat{\boldsymbol{J}}_{1,\delta_{n_{k,m_k}},2} \end{pmatrix} \\
\boldsymbol{H} = \begin{pmatrix} \boldsymbol{h}_{\delta_{n_{1,1}}} & & \\ & \ddots & \\ & & \boldsymbol{h}_{\delta_{n_{k,m_k}}} \end{pmatrix}, \boldsymbol{L} = \begin{pmatrix} \hat{\boldsymbol{J}}_{1,n_{1,1},2} & & \\ & \ddots & \\ & & \hat{\boldsymbol{J}}_{1,n_{k,m_k},2} \end{pmatrix} \\
\boldsymbol{P}_L = \begin{pmatrix} \gamma \cdot \hat{\boldsymbol{P}} & 0^{\hat{D} \times \hat{L}} & -\gamma \cdot \hat{\boldsymbol{V}} & 0^{\hat{D} \times t} & 0^{\hat{D} \times \dot{\ell}} & 0^{\hat{D} \times \dot{\ell}} \\ 0^{k \times \hat{L}} & 0^{k \times \hat{L}} & 0^{k \times t} & 0^{k \times t} & 0^{k \times \dot{\ell}} & 0^{k \times \dot{\ell}} \\ \boldsymbol{E} \cdot \boldsymbol{S} & \boldsymbol{E} \cdot \boldsymbol{S} & 0^{\hat{m} \times t} & 0^{\hat{m} \times t} & 0^{\hat{m} \times \dot{\ell}} & 0^{\hat{m} \times \dot{\ell}} \\ 0^{\hat{m} \times \hat{L}} & 0^{\hat{m} \times \hat{L}} & 0^{\hat{m} \times t} & 0^{\hat{m} \times t} & 0^{\hat{m} \times \dot{\ell}} & 0^{\hat{m} \times \dot{\ell}} \\ 0^{\hat{m} \times \hat{L}} & 0^{\hat{m} \times \hat{L}} & 0^{\hat{m} \times t} & 0^{\hat{m} \times t} & 0^{\hat{m} \times \dot{\ell}} & 0^{\hat{m} \times \dot{\ell}} \\ 0^{\ell_2 \times \hat{L}} & 0^{\ell_2 \times \hat{L}} & 0^{\ell_2 \times t} & 0^{\ell_2 \times t} & \gamma' \cdot \boldsymbol{M}^\intercal \cdot \hat{\boldsymbol{J}}_{\ell_1,q'-1,2} & 0^{\ell_2 \times \dot{\ell}} \end{pmatrix} \\
\boldsymbol{P}_R = \begin{pmatrix} 0^{\hat{D} \times 2\hat{m}} & 0^{\hat{D} \times 2\dot{n}} & 0^{\hat{D} \times 2\ddot{n}} & 0^{\hat{D} \times 2\ddot{n}} & 0^{\hat{D} \times 2\ddot{n}} \\ \gamma \cdot \boldsymbol{B} \cdot \hat{\boldsymbol{I}}_{\hat{m}} & 0^{k \times 2\dot{n}} & 0^{k \times 2\ddot{n}} & 0^{k \times 2\ddot{n}} & 0^{k \times 2\ddot{n}} \\ 0^{\hat{m} \times 2\hat{m}} & \boldsymbol{L} & 0^{\hat{m} \times 2\ddot{n}} & 0^{\hat{m} \times 2\ddot{n}} & 0^{\hat{m} \times 2\ddot{n}} \\ \hat{\boldsymbol{I}}_{\hat{m}} & \boldsymbol{H} & -\boldsymbol{L}' & 0^{\hat{m} \times 2\ddot{n}} & 0^{\hat{m} \times 2\ddot{n}} \\ 0^{\hat{m} \times 2\hat{m}} & \boldsymbol{H} & 0^{\hat{m} \times 2\ddot{n}} & -\boldsymbol{L}' & 0^{\hat{m} \times 2\ddot{n}} \\ 0^{\ell_2 \times 2\hat{m}} & 0^{\ell_2 \times 2\dot{n}} & 0^{\ell_2 \times 2\ddot{n}} & 0^{\ell_2 \times 2\ddot{n}} & 0^{\ell_2 \times 2\ddot{n}} \end{pmatrix} \\
\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_L & \boldsymbol{P}_R \end{pmatrix} \\
\\
\boldsymbol{n} = (n_{1,1}, \ldots, n_{k,m_k})^\intercal \\
\boldsymbol{v} = ((0^{\hat{D}})^\intercal \| \gamma \cdot \boldsymbol{c}^\intercal \| \boldsymbol{n}^\intercal \| (1^{\hat{m}})^\intercal \| (0^{\hat{m}})^\intercal \| \gamma' \cdot \boldsymbol{e}^\intercal)^\intercal
\end{cases}
$$

Then, we can transform Equation (14) as follows,

$$\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod \tilde{q}$$

and define VALID=VALID$_L$×VALID$_R$ that is a set of vectors of length $L = 2\hat{L} + 2t + 4\ell_1\delta_{q'-1} + 2\hat{m} + 2\dot{n} + 6\ddot{n}$, where

$$\begin{aligned}
\text{VALID}_L = \{&(\boldsymbol{w}_{1,1}^\intercal \| \ldots \| \boldsymbol{w}_{1,t}^\intercal \| \boldsymbol{w}_{2,1}^\intercal \| \ldots \| \boldsymbol{w}_{2,t}^\intercal \| \boldsymbol{w}_{3,1}^\intercal \| \ldots \| \boldsymbol{w}_{3,t}^\intercal \| \\
&\boldsymbol{w}_{4,1}^\intercal \| \ldots \| \boldsymbol{w}_{4,t}^\intercal \| \boldsymbol{w}_{5,1}^\intercal \| \ldots \| \boldsymbol{w}_{5,\ell_1}^\intercal \| \boldsymbol{w}_{6,1}^\intercal \| \ldots \| \boldsymbol{w}_{6,\ell_1}^\intercal)^\intercal \mid \\
&\forall i \in [1,t], \forall j \in [1,\ell_1] \text{ that } \rho(j) = i, \\
&(\boldsymbol{w}_{1,i} \in \text{VALID}_i \wedge \boldsymbol{w}_{2,i} \in 0^{L_i} \wedge \boldsymbol{w}_{3,i} = 1 \wedge \boldsymbol{w}_{4,i} = 0 \wedge \\
&\boldsymbol{w}_{5,j} \in \mathcal{D}_{\delta_{q'-1}} \wedge \boldsymbol{w}_{6,j} \in 0^{2\delta_{q'-1}}) \vee \\
&(\boldsymbol{w}_{1,i} \in 0^{L_i} \wedge \boldsymbol{w}_{2,i} \in \text{VALID}_i \wedge \boldsymbol{w}_{3,i} = 0 \wedge \boldsymbol{w}_{4,i} = 1 \wedge \\
&\boldsymbol{w}_{5,j} \in 0^{2\delta_{q'-1}} \wedge \boldsymbol{w}_{6,j} \in \mathcal{D}_{\delta_{q'-1}})\}
\end{aligned}$$

$$\begin{aligned}
\text{VALID}_R = \{&(\boldsymbol{w}_{1,1,1}^\intercal \| \ldots \| \boldsymbol{w}_{1,k,m_k}^\intercal \| \boldsymbol{w}_{2,1,1}^\intercal \| \ldots \| \boldsymbol{w}_{2,k,m_k}^\intercal \| \boldsymbol{w}_{3,1,1}^\intercal \| \ldots \| \\
&\boldsymbol{w}_{3,k,m_k}^\intercal \| \boldsymbol{w}_{4,1,1}^\intercal \| \ldots \| \boldsymbol{w}_{4,k,m_k}^\intercal \| \boldsymbol{w}_{5,1,1}^\intercal \| \ldots \| \boldsymbol{w}_{5,k,m_k}^\intercal)^\intercal \mid \\
&\forall i \in [1,k], \forall j \in [1,m_i], \boldsymbol{w}_{2,i,j} \in \mathcal{D}_{\delta_{n_{i,j}}} \wedge \boldsymbol{w}_{3,i,j} \in \mathcal{D}_{\delta_{\delta_{n_{i,j}}}} \wedge \\
&((\boldsymbol{w}_{1,i,j} = (0,1)^\intercal \wedge \boldsymbol{w}_{4,i,j} = 0^{2\delta_{\delta_{n_{i,j}}}} \wedge \boldsymbol{w}_{5,i,j} \in \mathcal{D}_{\delta_{\delta_{n_{i,j}}}}) \vee \\
&(\boldsymbol{w}_{1,i,j} = (1,0)^\intercal \wedge \boldsymbol{w}_{4,i,j} \in \mathcal{D}_{\delta_{\delta_{n_{i,j}}}} \wedge \boldsymbol{w}_{5,i,j} = 0^{2\delta_{\delta_{n_{i,j}}}}))\}
\end{aligned}$$

Obviously, $\boldsymbol{x}$ ∈VALID, and it can be easily verified that this statement is equivalent to the original statement indicated by Equation (13).
Then we need to specify the permutation used in the protocol. As each relation $\mathcal{R}_i$ can be proved under the abstract Stern's protocol, there exists $\mathcal{S}_i, \mathrm{T}^{(i)}$ that are valid for $\mathcal{R}_i$ for $i \in [1,t]$. Then we define

$$\begin{aligned}
\mathcal{S} = \{0,1\}^t \times \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_t \times (\{0,1\}^{\delta_{q'-1}})^{\ell_1} \times \{0,1\}^{\hat{m}} \times \{0,1\}^{\delta_{n_{1,1}}} \times \ldots \\
\times \{0,1\}^{\delta_{n_{k,m_k}}} \times \{0,1\}^{\delta_{\delta_{n_{1,1}}}} \times \ldots \times \{0,1\}^{\delta_{\delta_{n_{k,m_k}}}} \times \{0,1\}^{\delta_{\delta_{n_{1,1}}}} \times \ldots \times \{0,1\}^{\delta_{\delta_{n_{k,m_k}}}}
\end{aligned}$$

For

$$\begin{aligned}
\pi = (&\boldsymbol{b}_0, \pi_1, \pi_2, \ldots, \pi_t, \boldsymbol{b}_{1,1}, \ldots, \boldsymbol{b}_{1,\ell_1}, \boldsymbol{b}_{2,1,1}, \ldots, \boldsymbol{b}_{2,k,m_k}, \\
&\boldsymbol{b}_{3,1,1}, \ldots, \boldsymbol{b}_{3,k,m_k}, \boldsymbol{b}_{4,1,1}, \ldots, \boldsymbol{b}_{4,k,m_k}, \boldsymbol{b}_{5,1,1}, \ldots, \boldsymbol{b}_{5,k,m_k}) \in \mathcal{S}
\end{aligned}$$

where $\boldsymbol{b}_0 \in \{0,1\}^t$, for $i \in [1,t]$, $\pi_i \in \mathcal{S}_i$, for $j \in [1,\ell_1]$, $\boldsymbol{b}_{1,j} \in \{0,1\}^{\delta_{q'-1}}$, for $i \in [1, k]$, $j \in [1,m_i]$, $\boldsymbol{b}_{2,i,j} \in \{0,1\}$, $\boldsymbol{b}_{3,i,j} \in \{0,1\}^{\delta_{n_{i,j}}}$ $\boldsymbol{b}_{4,i,j} \in \{0,1\}^{\delta_{\delta_{n_{i,j}}}}$ $\boldsymbol{b}_{5,i,j} \in \{0,1\}^{\delta_{\delta_{n_{i,j}}}}$, and for

$$\begin{aligned}
\boldsymbol{w} = (&\boldsymbol{w}_{1,1}^\intercal, \ldots, \boldsymbol{w}_{1,t}^\intercal, \boldsymbol{w}_{2,1}^\intercal, \ldots, \boldsymbol{w}_{2,t}^\intercal, \boldsymbol{w}_{3,1}^\intercal, \ldots, \boldsymbol{w}_{3,t}^\intercal, \\
&\boldsymbol{w}_{4,1}^\intercal, \ldots, \boldsymbol{w}_{4,t}^\intercal, \boldsymbol{w}_{5,1}^\intercal, \ldots, \boldsymbol{w}_{5,\ell_1}^\intercal, \boldsymbol{w}_{6,1}^\intercal, \ldots, \boldsymbol{w}_{6,\ell_1}^\intercal, \\
&\boldsymbol{w}_{7,1,1}^\intercal, \ldots, \boldsymbol{w}_{7,k,m_k}^\intercal, \boldsymbol{w}_{8,1,1}^\intercal, \ldots, \boldsymbol{w}_{8,k,m_k}^\intercal, \boldsymbol{w}_{9,1,1}^\intercal, \ldots, \boldsymbol{w}_{9,k,m_k}^\intercal, \\
&\boldsymbol{w}_{10,1,1}^\intercal, \ldots, \boldsymbol{w}_{10,k,m_k}^\intercal, \boldsymbol{w}_{11,1,1}^\intercal, \ldots, \boldsymbol{w}_{11,k,m_k}^\intercal)^\intercal
\end{aligned}$$

where for $i \in [1,t]$, $\boldsymbol{w}_{1,i}, \boldsymbol{w}_{2,i}$ is of length L_i, and $\boldsymbol{w}_{3,i}, \boldsymbol{w}_{4,i}$ is of length 1, for $j \in [1, \ell_1]$, $\boldsymbol{w}_{5,j}, \boldsymbol{w}_{6,j}$ is of length $2\delta_{q'-1}$, and for $i \in [1,k]$, $j \in [1,m_i]$, $\boldsymbol{w}_{7,i,j}$ is of length 2, $\boldsymbol{w}_{8,i,j}$ is of length $2\delta_{n_{i,j}}$, and $\boldsymbol{w}_{9,i,j}$, $\boldsymbol{w}_{10,i,j}$, $\boldsymbol{w}_{11,i,j}$ is of length $2\delta_{\delta_{n_{i,j}}}$, we define

$$\begin{aligned}\mathtt{T}_\pi(\boldsymbol{w}) = (&\boldsymbol{w'}^\intercal_{1,1}, \ldots, \boldsymbol{w'}^\intercal_{1,t}, \boldsymbol{w'}^\intercal_{2,1}, \ldots, \boldsymbol{w'}^\intercal_{2,t}, \boldsymbol{w'}^\intercal_{3,1}, \ldots, \boldsymbol{w'}^\intercal_{3,t},\\ &\boldsymbol{w'}^\intercal_{4,1}, \ldots, \boldsymbol{w'}^\intercal_{4,t}, \boldsymbol{w'}^\intercal_{5,1}, \ldots, \boldsymbol{w'}^\intercal_{5,\ell_1}, \boldsymbol{w'}^\intercal_{6,1}, \ldots, \boldsymbol{w'}^\intercal_{6,\ell_1},\\ &\boldsymbol{w'}^\intercal_{7,1,1}, \ldots, \boldsymbol{w'}^\intercal_{7,k,m_k}, \boldsymbol{w'}^\intercal_{8,1,1}, \ldots, \boldsymbol{w'}^\intercal_{8,k,m_k}, \boldsymbol{w'}^\intercal_{9,1,1}, \ldots, \boldsymbol{w'}^\intercal_{9,k,m_k},\\ &\boldsymbol{w'}^\intercal_{10,1,1}, \ldots, \boldsymbol{w'}^\intercal_{10,k,m_k}, \boldsymbol{w'}^\intercal_{11,1,1}, \ldots, \boldsymbol{w'}^\intercal_{11,k,m_k})^\intercal\end{aligned}$$

that for $i \in [1,t]$, $\boldsymbol{w'}_{1,i} = \mathtt{T}^{(i)}_{\pi_i}(\boldsymbol{w}_{1+\boldsymbol{b}_0[i],i})$, $\boldsymbol{w'}_{2,i} = \mathtt{T}^{(i)}_{\pi_i}(\boldsymbol{w}_{2-\boldsymbol{b}_0[i],i})$, $\boldsymbol{w'}_{3,i} = \boldsymbol{w}_{3+\boldsymbol{b}_0[i],i}$, $\boldsymbol{w'}_{4,i} = \boldsymbol{w}_{4-\boldsymbol{b}_0[i],i}$, for $j \in [1,\ell_1]$, $\boldsymbol{w'}_{5,j} = \mathtt{F}_{\boldsymbol{b}_{1,j}}(\boldsymbol{w}_{5+\boldsymbol{b}_0[\rho(j)],j})$, $\boldsymbol{w'}_{6,j} = \mathtt{F}_{\boldsymbol{b}_{1,j}}(\boldsymbol{w}_{6-\boldsymbol{b}_0[\rho(j)],j})$, and for $i \in [1,k]$, $j \in [1,m_i]$, $\boldsymbol{w'}_{7,i,j} = \mathtt{F}_{\boldsymbol{b}_{2,i,j}}(\boldsymbol{w}_{7,i,j})$, $\boldsymbol{w'}_{8,i,j} = \mathtt{F}_{\boldsymbol{b}_{3,i,j}}(\boldsymbol{w}_{8,i,j})$, $\boldsymbol{w'}_{9,i,j} = \mathtt{F}_{\boldsymbol{b}_{4,i,j}}(\boldsymbol{w}_{9,i,j})$, $\boldsymbol{w'}_{10,i,j} = \mathtt{F}_{\boldsymbol{b}_{5,i,j}}(\boldsymbol{w}_{10+\boldsymbol{b}_{2,i,j},i,j})$, $\boldsymbol{w'}_{11,i,j} = \mathtt{F}_{\boldsymbol{b}_{5,i,j}}(\boldsymbol{w}_{11-\boldsymbol{b}_{2,i,j},i,j})$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

D Concrete ZKAoKs for wPRF with Efficient Protocols

In this section, we give constructions of ZKAoKs for $y = F_k(x)$ in different cases, including the case that k is hidden, the case that k, y are hidden, the case that k, x are hidden, and the case that k, x, y are hidden. Recall that this is equal to prvoing in these cases that for a key, input, output tuple $(\boldsymbol{s}, \boldsymbol{A}, \boldsymbol{y}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_p^m$, there exists a vector $e \in [-\beta,\beta]^m$ that $\boldsymbol{A}\cdot\boldsymbol{s} + \boldsymbol{e} = \gamma\cdot\boldsymbol{y} \mod q$.

1. Proving $y = F_k(x)$ ***with hidden*** k**.** First, we consider the basic case that only the secret key needs to be hidden. In this case, the common input includes $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ and $\boldsymbol{y} \in \mathbb{Z}_p^m$, and the prover's secret input includes $\boldsymbol{s} \in \mathbb{Z}_q^n$ and $\boldsymbol{e} \in [-\beta,\beta]^m$. To complete the proof, we employ the general framework shown in [LLM+16a]. First we decompose and extend Equation (7) and define $\hat{\boldsymbol{s}} = DecEnc2_{n,q-1}(\boldsymbol{s})$, $\hat{\boldsymbol{e}} = DecExt3_{m,\beta}(\boldsymbol{e})$, $\boldsymbol{x} = (\hat{\boldsymbol{s}}^\intercal \| \hat{\boldsymbol{e}}^\intercal)^\intercal$. We also define $\boldsymbol{P} = (\boldsymbol{A}\cdot\hat{\boldsymbol{J}}_{\boldsymbol{n,q-1,2}}, \hat{\boldsymbol{K}}_{\boldsymbol{m,\beta,3}})$ and $\boldsymbol{v} = \gamma\cdot\boldsymbol{y}$. Then, we can transform Equation (7) to

$$\boldsymbol{P}\cdot\boldsymbol{x} = \boldsymbol{v} \mod q$$

and define VALID= $\mathcal{D}_{n\delta_{q-1}} \times \mathcal{B}^3_{m\delta_\beta}$ that is a set of vectors of length $L = 2n\delta_{q-1} + 3m\delta_\beta$. Obviously, $\boldsymbol{x} \in$VALID. It can be easily verified that this statement is equivalent to the original statement in this case, namely $y = F_k(x)$ with hidden k and public x, y.

Then we need to specify the permutation used in the protocol. First, we define $\mathcal{S} = \{0,1\}^{n\delta_{q-1}} \times \mathcal{P}_{3m\delta_\beta}$. Next, for $\pi = (\boldsymbol{b_1}, \pi_2) \in \mathcal{S}$ where $\boldsymbol{b_1} \in \{0,1\}^{n\delta_{q-1}}$ and $\pi_2 \in \mathcal{P}_{3m\delta_\beta}$, and $\boldsymbol{w} = (\boldsymbol{w_1}^\intercal \| \boldsymbol{w_2}^\intercal)^\intercal$ where $\boldsymbol{w_1}$ is of length $2n\delta_{q-1}$ and $\boldsymbol{w_2}$ is of length $3m\delta_\beta$, we define $\mathtt{T}_\pi(\boldsymbol{w}) = (\mathtt{F}_{\boldsymbol{b_1}}(\boldsymbol{w_1})^\intercal, \pi_2(\boldsymbol{w_2})^\intercal)^\intercal$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

2. Proving $y = F_k(x)$ ***with hidden*** k, y**.** Then we consider a slightly different case that both the secret key and the output need to be hidden. In this case, the common input includes $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ and the prover's secret input includes $\boldsymbol{s} \in \mathbb{Z}_q^n$, $\boldsymbol{e} \in [-\beta, \beta]^m$, and $\boldsymbol{y} \in \mathbb{Z}_p^m$. We also employ the general framework shown in [LLM+16a] to handle this proof. First we decompose and extend Equation (7) and define $\hat{\boldsymbol{s}} = DecEnc2_{n,q-1}(\boldsymbol{s})$, $\hat{\boldsymbol{e}} = DecExt3_{m,\beta}(\boldsymbol{e})$, $\hat{\boldsymbol{y}} = DecEnc2_{m,p-1}(\boldsymbol{y})$, $\boldsymbol{x} = (\hat{\boldsymbol{s}}^\top \| \hat{\boldsymbol{e}}^\top \| \hat{\boldsymbol{y}}^\top)^\top$. We also define $\boldsymbol{P} = (\boldsymbol{A} \cdot \hat{\boldsymbol{J}}_{\boldsymbol{n,q-1,2}},\ \hat{\boldsymbol{K}}_{\boldsymbol{m,\beta,3}},\ -\gamma \cdot \hat{\boldsymbol{J}}_{\boldsymbol{m,p-1,2}})$ and $\boldsymbol{v} = 0^m$. Then, we can transform Equation (7) to

$$\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q$$

and define VALID= $\mathcal{D}_{n\delta_{q-1}} \times \mathcal{B}^3_{m\delta_\beta} \times \mathcal{D}_{m\delta_{p-1}}$ that is a set of vectors of length $L = 2n\delta_{q-1} + 3m\delta_\beta + 2m\delta_{p-1}$. Obviously, $\boldsymbol{x} \in$VALID. It can be easily verified that this statement is equivalent to the original statement in this case, namely $y = F_k(x)$ with hidden k, y and public x.

Then we need to specify the permutation used in the protocol. First, we define $\mathcal{S} = \{0,1\}^{n\delta_{q-1}} \times \mathcal{P}_{3m\delta_\beta} \times \{0,1\}^{m\delta_{p-1}}$. Next, for $\pi = (\boldsymbol{b_1}, \pi_2, \boldsymbol{b_3}) \in \mathcal{S}$, where $\boldsymbol{b_1} \in \{0,1\}^{n\delta_{q-1}}$, $\pi_2 \in \mathcal{P}_{3m\delta_\beta}$, and $\boldsymbol{b_3} \in \{0,1\}^{m\delta_{p-1}}$, and $\boldsymbol{w} = (\boldsymbol{w_1}^\top \| \boldsymbol{w_2}^\top \| \boldsymbol{w_3}^\top)^\top$, where $\boldsymbol{w_1}$ is of length $2n\delta_{q-1}$, $\boldsymbol{w_2}$ is of length $3m\delta_\beta$, and $\boldsymbol{w_3}$ is of length $2m\delta_{p-1}$, we define $\mathtt{T}_\pi(\boldsymbol{w}) = (\mathtt{F}_{\boldsymbol{b_1}}(\boldsymbol{w_1})^\top, \pi_2(\boldsymbol{w_2})^\top, \mathtt{F}_{\boldsymbol{b_3}}(\boldsymbol{w_3})^\top)^\top$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

3. Proving $y = F_k(x)$ ***with hidden*** k, x**.** Next, we consider the case that both the secret key and the input need to be hidden. In this case, the common input includes $\boldsymbol{y} \in \mathbb{Z}_p^m$, and the prover's secret input includes $\boldsymbol{s} \in \mathbb{Z}_q^n$, $\boldsymbol{e} \in [-\beta, \beta]^m$, and $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$. To conduct the proof with a hidden $\boldsymbol{A}$, we exploit the technique in [LLM+16b], which provides a method to prove LWE relation with hidden matrices under the abstract Stern's protocol[LLM+16a]. More precisely, we define $\dot{k} = \delta_{q-1}$, and define

$$\begin{cases} \hat{\boldsymbol{a}} = DecEnc2_{mn,q-1}(M2V(\boldsymbol{A})), \hat{\boldsymbol{s}} = DecEnc2_{n,q-1}(\boldsymbol{s}) \\ \hat{\boldsymbol{e}} = DecExt3_{m,\beta}(\boldsymbol{e}), \boldsymbol{z} = Expand_{mn\dot{k},n\dot{k},m\dot{k},\dot{k}}(\hat{\boldsymbol{a}}, \hat{\boldsymbol{s}}) \\ \boldsymbol{x} = (\hat{\boldsymbol{a}}^\top \| \hat{\boldsymbol{s}}^\top \| \hat{\boldsymbol{e}}^\top \| \boldsymbol{z}^\top)^\top \\ \\ \boldsymbol{P} = (0^{m\times(2mn\dot{k}+2n\dot{k})}, \hat{\boldsymbol{K}}_{\boldsymbol{m,\beta,3}}, \boldsymbol{Q}_{m,n,q-1,q-1}) \\ \\ \boldsymbol{v} = \gamma \cdot \boldsymbol{y}. \end{cases}$$

Then, we can transform Equation (7) to

$$\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q$$

and define

$$\begin{aligned} \text{VALID}=\{(\boldsymbol{w_1}^\top \| \boldsymbol{w_2}^\top \| \boldsymbol{w_3}^\top \| \boldsymbol{w_4}^\top)^\top \mid \boldsymbol{w_1} \in \mathcal{D}_{mn\dot{k}} \wedge \boldsymbol{w_2} \in \mathcal{D}_{n\dot{k}} \\ \wedge\, \boldsymbol{w_3} \in \mathcal{B}^3_{m\delta_\beta} \wedge \boldsymbol{w_4} = Expand_{mn\dot{k},n\dot{k},m\dot{k},\dot{k}}(\boldsymbol{w_1}, \boldsymbol{w_2})\} \end{aligned}$$

that is a set of vectors of length $L = 2mn\dot{k} + 2n\dot{k} + 3m\delta_\beta + 4mn\dot{k}^2$. Obviously, $\boldsymbol{x} \in$VALID. It can be easily verified that this statement is equivalent to the original statement in this case, namely $y = F_k(x)$ with hidden k, x and public y.
Then we need to specify the permutation used in the protocol. First, we define $\mathcal{S} = \{0,1\}^{mn\dot{k}} \times \{0,1\}^{n\dot{k}} \times \mathcal{P}_{3m\delta_\beta}$. Next, for $\pi = (\boldsymbol{b_1}, \boldsymbol{b_2}, \pi_3) \in \mathcal{S}$, where $\boldsymbol{b_1} \in \{0,1\}^{mn\dot{k}}$, $\boldsymbol{b_2} \in \{0,1\}^{n\dot{k}}$, and $\pi_3 \in \mathcal{P}_{3m\delta_\beta}$, and $\boldsymbol{w} = (\boldsymbol{w_1}^\intercal \| \boldsymbol{w_2}^\intercal \| \boldsymbol{w_3}^\intercal \| \boldsymbol{w_4}^\intercal)^\intercal$, where $\boldsymbol{w_1}$ is of length $2mn\dot{k}$, $\boldsymbol{w_2}$ is of length $2n\dot{k}$, $\boldsymbol{w_3}$ is of length $3m\delta_\beta$, and $\boldsymbol{w_4}$ is of length $4mn\dot{k}^2$, we define $\mathtt{T}_\pi(\boldsymbol{w}) = (\mathtt{F}_{\boldsymbol{b_1}}(\boldsymbol{w_1})^\intercal, \mathtt{F}_{\boldsymbol{b_2}}(\boldsymbol{w_2})^\intercal, \pi_3(\boldsymbol{w_3})^\intercal, \mathtt{P}_{\boldsymbol{b_1},\boldsymbol{b_2},m\dot{k},\dot{k}}(\boldsymbol{w_4})^\intercal)^\intercal$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

4. Proving $y = F_k(x)$ ***with hidden*** k, x, y**.** Finally, we consider the case that the secret key, the input and the output all need to be hidden. In this case, there is no common input, and the prover's secret input includes $\boldsymbol{s} \in \mathbb{Z}_q^n$, $\boldsymbol{e} \in [-\beta, \beta]^m$, $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$, and $\boldsymbol{y} \in \mathbb{Z}_p^m$. We employ techniques that has been used in the last two cases to construct this proof. More precisely, we define $\dot{k} = \delta_{q-1}$, and define

$$\begin{cases} \hat{\boldsymbol{a}} = DecEnc2_{mn,q-1}(M2V(\boldsymbol{A})), \hat{\boldsymbol{s}} = DecEnc2_{n,q-1}(\boldsymbol{s}), \hat{\boldsymbol{e}} = DecExt3_{m,\beta}(\boldsymbol{e}) \\ \boldsymbol{z} = Expand_{mn\dot{k},nk,mk,\dot{k}}(\hat{\boldsymbol{a}}, \hat{\boldsymbol{s}}), \hat{\boldsymbol{y}} = DecEnc2_{m,p-1}(\boldsymbol{y}) \\ \boldsymbol{x} = (\hat{\boldsymbol{a}}^\intercal \| \hat{\boldsymbol{s}}^\intercal \| \hat{\boldsymbol{e}}^\intercal \| \boldsymbol{z}^\intercal \| \hat{\boldsymbol{y}}^\intercal)^\intercal \\ \\ \boldsymbol{P} = (0^{m \times (2mn\dot{k} + 2n\dot{k})},\ \hat{\boldsymbol{K}}_{\boldsymbol{m,\beta,3}},\ \boldsymbol{Q}_{m,n,q-1,q-1},\ -\gamma \cdot \hat{\boldsymbol{J}}_{\boldsymbol{m,p-1,2}}) \\ \\ \boldsymbol{v} = 0^m. \end{cases}$$

Then, we can transform Equation (7) to

$$\boldsymbol{P} \cdot \boldsymbol{x} = \boldsymbol{v} \mod q$$

and define

$$\begin{aligned} \text{VALID}=\{(\boldsymbol{w_1}^\intercal \| \boldsymbol{w_2}^\intercal \| \boldsymbol{w_3}^\intercal \| \boldsymbol{w_4}^\intercal \| \boldsymbol{w_5}^\intercal)^\intercal \mid \boldsymbol{w_1} \in \mathcal{D}_{mn\dot{k}} \wedge \boldsymbol{w_2} \in \mathcal{D}_{n\dot{k}} \\ \wedge \boldsymbol{w_3} \in \mathcal{B}^3_{m\delta_\beta} \wedge \boldsymbol{w_4} = Expand_{mn\dot{k},nk,mk,\dot{k}}(\boldsymbol{w_1}, \boldsymbol{w_2}) \wedge \boldsymbol{w_5} \in \mathcal{D}_{m\delta_{p-1}}\}. \end{aligned}$$

that is a set of vectors of length $L = 2mn\dot{k}+2n\dot{k}+3m\delta_\beta+4mn\dot{k}^2+2m\delta_{p-1}$. Obviously, $\boldsymbol{x} \in$VALID. It can be easily verified that this statement is equivalent to the original statement in this case, namely $y = F_k(x)$ with hidden k, x, y.
Then we need to specify the permutation used in the protocol. First, we define $\mathcal{S} = \{0,1\}^{mn\dot{k}} \times \{0,1\}^{n\dot{k}} \times \mathcal{P}_{3m\delta_\beta} \times \{0,1\}^{m\delta_{p-1}}$. Next, for $\pi = (\boldsymbol{b_1}, \boldsymbol{b_2}, \pi_3, \boldsymbol{b_4}) \in \mathcal{S}$, where $\boldsymbol{b_1} \in \{0,1\}^{mn\dot{k}}$, $\boldsymbol{b_2} \in \{0,1\}^{n\dot{k}}$, $\pi_3 \in \mathcal{P}_{3m\delta_\beta}$, and $\boldsymbol{b_4} \in \{0,1\}^{m\delta_{p-1}}$, and $\boldsymbol{w} = (\boldsymbol{w_1}^\intercal \| \boldsymbol{w_2}^\intercal \| \boldsymbol{w_3}^\intercal \| \boldsymbol{w_4}^\intercal \| \boldsymbol{w_5}^\intercal)^\intercal$, where $\boldsymbol{w_1}$ is of length $2mn\dot{k}$, $\boldsymbol{w_2}$ is of length $2n\dot{k}$, $\boldsymbol{w_3}$ is of length $3m\delta_\beta$, $\boldsymbol{w_4}$ is of length $4mn\dot{k}^2$, and $\boldsymbol{w_5}$ is of length $2m\delta_{p-1}$, we define $\mathtt{T}_\pi(\boldsymbol{w}) = (\mathtt{F}_{\boldsymbol{b_1}}(\boldsymbol{w_1})^\intercal, \mathtt{F}_{\boldsymbol{b_2}}(\boldsymbol{w_2})^\intercal, \pi_3(\boldsymbol{w_3})^\intercal, \mathtt{P}_{\boldsymbol{b_1},\boldsymbol{b_2},m\dot{k},\dot{k}}(\boldsymbol{w_4})^\intercal, \mathtt{F}_{\boldsymbol{b_5}}(\boldsymbol{w_5})^\intercal)^\intercal$. It is not hard to check that $\mathcal{S}$ and $\mathtt{T}$ are valid.

E Formal Definitions of Applications

E.1 Definition of The Linkable Ring Signature

In this section, we recall the syntax and the security model of linkable ring signature, which is refined in [ACST06]. Note that there is a few small mistakes in their security model, and we will correct them here. A linkable ring signature consists of five algorithms:

- **Setup.** On input a security parameter 1^λ, the setup algorithm outputs the public parameter *param* for the scheme, which is also set implicity as the input for the following four algorithms.
- **KeyGen.** The key generation algorithm outputs a secret key/public key pair $(sk, pk) \in \mathcal{SK} \times \mathcal{PK}$, where we use $\mathcal{SK}$ and $\mathcal{PK}$ to denote the secret key space and the public key space respectively.
- **Sign.** On input a message $\mathfrak{m}$, a polynomial-size set $\mathcal{R}$ of public keys, and a secret key sk whose corresponding public key is in $\mathcal{R}$, the signing algorithm outputs a signature σ.
- **Verify.** On input a message $\mathfrak{m}$, a polynomial-size set $\mathcal{R}$ of public keys, and a signature σ, the verification algorithm outputs a bit indicating whether the signature is acceptable.
- **Link.** On input two signatures σ_0 and σ_1, the linking algorithm outputs a bit indicating whether the two signature are signed by the same user.

Correctness of the linkable ring signature scheme requires that an honestly signed signature should pass the verification and that two honestly generated signatures can be linked iff they are signed by the same user. More formally, we require that:

- **Verification Correctness.** Let $param \leftarrow Setup(1^\lambda)$, $(sk, pk) \leftarrow KeyGen()$. Then for any message $\mathfrak{m}$, any polynomial-size set $\mathcal{R} \in 2^{\mathcal{PK}}$ containing pk, we have $\Pr[Verif(\mathfrak{m}, \mathcal{R}, Sign(\mathfrak{m}, \mathcal{R}, sk)) = 1] \geq 1 - negl(\lambda)$.
- **Linking Correctness.** Let $param \leftarrow Setup(1^\lambda)$, $(sk_1, pk_1) \leftarrow KeyGen()$, $(sk_2, pk_2) \leftarrow KeyGen()$. Then for any message $\mathfrak{m}_1, \mathfrak{m}_2$, any polynomial-size set $\mathcal{R}_1$, $\mathcal{R}_1' \in 2^{\mathcal{PK}}$ containing pk_1, and any polynomial-size set $\mathcal{R}_2 \in 2^{\mathcal{PK}}$ containing pk_2, we have $\Pr[Link(Sign(\mathfrak{m}_1, \mathcal{R}_1, sk_1), Sign(\mathfrak{m}_2, \mathcal{R}_1', sk_1)) = 1] \geq 1 - negl(n)$ and $\Pr[Link(Sign(\mathfrak{m}_1, \mathcal{R}_1, sk_1), Sign(\mathfrak{m}_2, \mathcal{R}_2, sk_2)) = 0] \geq 1 - negl(n)$.

Security of the linkable ring signature scheme requires that the scheme is unforgeable, linkable-anonymous w.r.t. adversarially-chosen keys, linkable w.r.t. adversarially-chosen keys, and non-slanderable w.r.t. adversarially-chosen keys, which is formally defined as follows.

Unforgeability. A linkable ring signature scheme is unforgeable if for any probabilistic polynomial time adversary $\mathcal{A}$ and for any polynomial $n(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible in λ.

1. In the beginning, the challenger generates $param \leftarrow Setup(1^\lambda)$ and $(sk_i, pk_i) \leftarrow KeyGen(param; \omega_i)$ for $i \in [1, n]$, where ω_i is the randomness used in the generation of the ith key pair. Then it sends the public parameter *param* and the set $\mathcal{S} = \{pk_i\}_{i=1}^n$ to $\mathcal{A}$. It also initilizes two empty sets $\mathcal{SO}$ and $\mathcal{CO}$.
2. Then $\mathcal{A}$ is allowed to access the following two oracles:

- *The Signing Oracle.* On input an integer $j \in [1, n]$, a message $\mathfrak{m}$ and a ring $\mathcal{R} \subseteq \mathcal{PK}$ containing pk_j, the challenger returns $\sigma \leftarrow Sign(\mathfrak{m}, \mathcal{R}, sk_j)$ and puts $(m, \mathcal{R}, \sigma)$ into $\mathcal{SO}$.
- *The Corrupt Oracle.* On input an integer $j \in [1, n]$, the challenger returns ω_j and puts pk_j into $\mathcal{CO}$.

3. Finally, $\mathcal{A}$ outputs $(\mathfrak{m}^*, \mathcal{R}^*, \sigma^*)$, and succeeds if $Verify(\mathfrak{m}^*, \mathcal{R}^*, \sigma^*) = 1$, $(\mathfrak{m}^*, \mathcal{R}^*, \sigma^*) \notin \mathcal{SO}$, $\mathcal{R}^* \subseteq \mathcal{S}$, $\mathcal{R}^* \cap \mathcal{CO} = \emptyset$.

Linkable-Anonymity w.r.t. adversarially-chosen keys. A linkable ring signature scheme is linkable anonymous w.r.t. adversarially-chosen keys if for any probabilistic polynomial time admissible adversary $\mathcal{A}$ and for any polynomial $n(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligibly close to 1/2.

1. In the beginning, the challenger generates $param \leftarrow Setup(1^\lambda)$ and $(sk_i, pk_i) \leftarrow KeyGen(param; \omega_i)$ for $i \in [1, n]$, where ω_i is the randomness used in the generation of the ith key pair. Then it sends the public parameter $param$ and the set $\mathcal{S} = \{pk_i\}_{i=1}^n$ to $\mathcal{A}$. It also initilizes an empty set $\mathcal{Q}$.
2. Then $\mathcal{A}$ is allowed to access the following two oracles:
   - *The Signing Oracle.* On input an integer $j \in [1, n]$, a message $\mathfrak{m}$ and a ring $\mathcal{R} \subseteq \mathcal{PK}$ containing pk_j, the challenger returns $\sigma \leftarrow Sign(\mathfrak{m}, \mathcal{R}, sk_j)$ and puts j into $\mathcal{Q}$.
   - *The Corrupt Oracle.* On input an integer $j \in [1, n]$, the challenger returns ω_j and puts j into $\mathcal{Q}$.
3. Next, $\mathcal{A}$ submits a message $\mathfrak{m}^*$, two distinct integers $i_0^*, i_1^* \in [1, n]$, and a ring $\mathcal{R}^* \subseteq \mathcal{PK}$ that $pk_{i_0^*}, pk_{i_1^*} \in \mathcal{R}^*$. Then, the challenger choose a random bit $b \in \{0, 1\}$, and returns a signature $\sigma^* \leftarrow Sign(\mathfrak{m}^*, \mathcal{R}^*, sk_{i_b^*})$.
4. After receiving the challenge signature σ^*, $\mathcal{A}$ is further allowed to aceess the signing oralce and the corrupt oracle as in the second phase.
5. Finally, $\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$. Here, we say an adversary $\mathcal{A}$ being admissible if $i_0 \notin \mathcal{Q}$ and $i_1 \notin \mathcal{Q}$.

Linkability w.r.t. adversarially-chosen keys. A linkable ring signature scheme is linkable w.r.t. adversarially-chosen keys if for any probabilistic polynomial time adversary $\mathcal{A}$ and for any polynomial $n(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible.

1. In the beginning, the challenger generates $param \leftarrow Setup(1^\lambda)$ and $(sk_i, pk_i) \leftarrow KeyGen(param; \omega_i)$ for $i \in [1, n]$, where ω_i is the randomness used in the generation of the ith key pair. Then it sends the public parameter $param$ and the set $\mathcal{S} = \{pk_i\}_{i=1}^n$ to $\mathcal{A}$. It also initilizes two empty sets $\mathcal{SO}$ and $\mathcal{CO}$.
2. Then $\mathcal{A}$ is allowed to access the following two oracles:
   - *The Signing Oracle.* On input an integer $j \in [1, n]$, a message $\mathfrak{m}$ and a ring $\mathcal{R} \subseteq \mathcal{PK}$ containing pk_j, the challenger returns $\sigma \leftarrow Sign(\mathfrak{m}, \mathcal{R}, sk_j)$ and puts $(m, \mathcal{R}, \sigma)$ into $\mathcal{SO}$.
   - *The Corrupt Oracle.* On input an integer $j \in [1, n]$, the challenger returns ω_j and puts pk_j into $\mathcal{CO}$.
3. Finally, $\mathcal{A}$ outputs $(\mathfrak{m}_1^*, \mathcal{R}_1^*, \sigma_1^*)$ and $(\mathfrak{m}_2^*, \mathcal{R}_2^*, \sigma_2^*)$, and succeeds if $Verify(\mathfrak{m}_1^*, \mathcal{R}_1^*, \sigma_1^*) = 1$, $Verify(\mathfrak{m}_2^*, \mathcal{R}_2^*, \sigma_2^*) = 1$, $Link(\sigma_1^*, \sigma_2^*) = 0$, $(\mathfrak{m}_1^*, \mathcal{R}_1^*, \sigma_1^*) \notin \mathcal{SO}$, $(\mathfrak{m}_2^*, \mathcal{R}_2^*, \sigma_2^*) \notin \mathcal{SO}$, and $| (\mathcal{R}_1^* \cup \mathcal{R}_2^*) \cap \mathcal{CO} | + | (\mathcal{R}_1^* \cup \mathcal{R}_2^*) \backslash \mathcal{S} | \leq 1$.

Non-slanderability w.r.t. adversarially-chosen keys. A linkable ring signature scheme is non-slanderable w.r.t. adversarially-chosen keys if for any probabilistic polynomial time adversary $\mathcal{A}$ and for any polynomial $n(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible.

1. In the beginning, the challenger generates $param \leftarrow Setup(1^\lambda)$ and $(sk_i, pk_i) \leftarrow KeyGen(param; \omega_i)$ for $i \in [1, n]$, where ω_i is the randomness used in the generation of the ith key pair. Then it sends the public parameter $param$ and the set $\mathcal{S} = \{pk_i\}_{i=1}^n$ to $\mathcal{A}$. It also initilizes two empty sets $\mathcal{SO}, \mathcal{CO}$.
2. Then $\mathcal{A}$ is allowed to access the following two oracles:
   - *The Signing Oracle.* On input an integer $j \in [1, n]$, a message $\mathfrak{m}$ and a ring $\mathcal{R} \subseteq \mathcal{PK}$ containing pk_j, the challenger returns $\sigma \leftarrow Sign(\mathfrak{m}, \mathcal{R}, sk_j)$ and puts $(j, \mathfrak{m}, \mathcal{R}, \sigma)$ into $\mathcal{SO}$.
   - *The Corrupt Oracle.* On input an integer $j \in [1, n]$, the challenger returns ω_j and puts j into $\mathcal{CO}$.
3. Finally, $\mathcal{A}$ outputs $(\hat{j}, \hat{\sigma}, \mathfrak{m}^*, \mathcal{R}^*, \sigma^*)$, and succeeds if $Verify(\mathfrak{m}^*, \mathcal{R}^*, \sigma^*) = 1$, $Link(\hat{\sigma}, \sigma^*) = 1$, $(\hat{j}, *, *, \hat{\sigma}) \in \mathcal{SO}$, $(\hat{j}, \mathfrak{m}^*, \mathcal{R}^*, \sigma^*) \notin \mathcal{SO}$, $\hat{j} \notin \mathcal{CO}$.

E.2 Definition of The k-times Anonymous Authentication

In this section, we recall the syntax and the security model of k-times anonymous authentication, which is defined in [TFS04]. A k-times anonymous authentication involves three types of entities, namely, the group manager, the application providers, and the users. It also consists of five protocols (algorithms):

- **Setup.** In the setup protocol, the group manager generates its group public key and group secret key, and publish the group public key.
- **Join.** The join protocol is run between a user and the group manager, and after the procedure, the user, who is called a group member now, obtains a member public key and a member secret key.
- **Bound Announcement.** In the bound announcement protocol, an application provider AP publish its identity ID_{AP} and the upper bound k_{AP} for each user to access its service.
- **Authentication.** The authentication protocol is run between a user and an application provider. The application provider accepts the user iff the user is a legally registered group member and has not accessed its service more than $k_{AP} - 1$ times. The transcript of the protocol will be recorded into the authentication log by the application provider no matter whether the user is accepted.
- **Public Tracing.** On input an authentication log and an identification list, which records the map between user identifications and member public keys, the public tracing algorithm outputs a subset of $\mathcal{U} \cup \{GM\}$ to indicate the cheaters, where $\mathcal{U}$ is the set of all users and GM is the group manager.

Correctness of the k-times anonymous authentication requires that an honest group member can be accepted in authentication with an honest application provider AP if he has not accessed the service of AP more than $k_{AP} - 1$ times.

Security of the k-times anonymous authentication requires that the system is Anonymous, detectable, exculpable for users and exculpable for the group manager, which can be formally defined by the following experiments. In these experiments, the adversary is allowed to corrupt a few parties, and:

- If the adversary colludes with the group manager, the adversary can maliciously execute the setup procedure and the join porotocol on behalf of the group manager.
- If the adversary colludes with a user i, the adversary can maliciously execute the join protocol and the authentication protocol on behalf of the user i.
- If the adversary colludes with an application provider AP, the adversary can choose the public information (ID_{AP}, k_{AP}) of AP and can maliciously execute the authentication protocol on behalf of the application provider AP. Moreover, the adversary can use different information (ID_{AP}, k_{AP}) for each authentication.

It is also allowed to query some of the following oracles in each experiment:

- The oracle $O_{LIST}[\mathcal{CE}]$: The list oracle is parameterized with a set $\mathcal{CE}$, which is a subset of the set $\mathcal{U} \cup \{GM\}$. It also matains a list $\mathcal{LIST}$, which is empty in the beginning. On input a command $c \in \{0, 1, 2\}$ which represents the request of querying the public key of a user, adding a new public key and deleting a public key respectively, an identity i of a user, and a user public key mpk, the oracle proceeds as follows:
  1. If $c = 0$, mpk is empty, and $\exists mpk'$ s.t. $(i, mpk') \in \mathcal{LIST}$, the oracle returns mpk'.
  2. If $c = 1$, $i \in \mathcal{CE}$, and $\nexists mpk'$ s.t. $(i, mpk') \in \mathcal{LIST}$, the oracle set the list $\mathcal{LIST} = \mathcal{LIST} \cup \{(i, mpk)\}$ and returns $\perp$.
  3. If $c = 2$, $GM \in \mathcal{CE}$, mpk is empty, and $\exists mpk'$ s.t. $(i, mpk') \in \mathcal{LIST}$, the oracle set the list $\mathcal{LIST} = \mathcal{LIST} \backslash \{(i, mpk')\}$ and returns $\perp$.
  4. Otherwise, the oracle aborts and returns $\perp$.
- The oracle $O_{Query}[b, gpk, (i_0, i_1), (ID, k)]$: The query oracle is parameterized with a bit b, a group manager public key gpk, two user identities (i_0, i_1), and the public information (ID, k) of an application provider. On input a bit d and a string M, the oracle returns $O_{Auth-U}[gpk](i_{b \oplus d}, (ID, k), M)$.
- The oracle $O_{Join-U}[gpk]$: On input an identity i, the oracle executes the joining protocol on behalf of the honest user i with the adversary.
- The oracle $O_{Auth-U}[gpk]$: On input an identity i, the public information (ID, k) of an application provider, and a challenge message M, the oracle executes the authentication protocol on behalf of the honest user i with the adversary, who represents an application provider with public information (ID, k) and challenge message M.
- The oracle $O_{Join-GM}[gpk, gsk]$: On input an identity i, and a joining message M, the oracle executes the joining protocol on behalf of the honest group manager, whose group key pair is (gpk, gsk), with the adversary, who represents a user i with joining message M.
- The oracle $O_{Auth-AP}[gpk]$: On input the public information (ID_{AP}, k_{AP}) of an application provider AP, the oracle executes the authentication protocol on behalf of the honest application provider AP with the adversary, who represents a group member.

Besides, the adversary is only allowed to execute many authentication procedures sequentially, but it is allowed to execute many joining procedures concurrently, i.e. we require the k-times anonymous authentication to support concurrent enronment.

Anonymity. A k-times anonymous authentication is anonymous if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligibly close to $1/2$.

1. In the beginning, the challenger chooses a bit b, two target users i_0, i_1, and an application provider AP with public information (ID^*, k^*). Then it publish (i_0, i_1) and (ID^*, k^*)
2. Then, the adversary $\mathcal{A}$ corrupts the group manager, all application providers (including AP), and all users except the two target users i_0, i_1. It also generates a (malicious) group public key gpk.
3. Then $\mathcal{A}$ is allowed to access the oracle $O_{List}[\mathcal{U} \cup \{GM\} - \{i_0, i_1\}]$, $O_{Join-U}[gpk]$, $O_{Auth-U}[gpk]$, and $O_{Query}[b, gpk, (i_0, i_1), (ID^*, k^*)]$ multiple times in any order, but with the restriction that it can only access the oracle O_{Query} one time of the form $(d, *)$ for each queried $d \in \{0, 1\}$, and can only access the oracle $O_{Auth-U}[gpk]$ $k-1$ times of the form $(i, (ID^*, M^*))$ for each queried $i \in \{i_0, i_1\}$.
4. Finally, $\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$.

Detectability. A k-times anonymous authentication is detectable if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligible.

1. In the beginning, the challenger generates the group key pair (gpk, gsk) by running the setup procedure, and sends gpk to the adversary $\mathcal{A}$.
2. Then, the adversary $\mathcal{A}$, who controls all users in the system, is allowed to access the oracle $O_{List}[\mathcal{U}]$, $O_{Join-GM}[gpk, gsk]$, and $O_{Auth-AP}[gpk]$ multiple times in any order. Here, whenever the adversary queries the oracle $O_{Join-GM}[gpk, gsk]$ with an identity i and a user public key mpk, it should also query the oracle $O_{List}[\mathcal{U}]$ with input $(1, i, mpk)$.
3. Finally, the challenger checks authentication logs of applications providers. The adversary $\mathcal{A}$ succeeds in the experiment if there exists an application provider AP with public information ID_{AP}, k_{AP} and public log $\mathcal{LOG}_{AP}$ s.t. the number of valid items in $\mathcal{LOG}_{AP}$ is larger than k_{AP} times the number of items in $\mathcal{LIST}$ and the result of running the public tracing algorithm on input $(gpk, \mathcal{LOG}_{AP}, \mathcal{LIST})$ is the empty set.

Exculpability for users. A k-times anonymous authentication is exculpable for users if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligible.

1. In the beginning, the challenger chooses a target users i^* and publish it.
2. Then, the adversary $\mathcal{A}$ corrupts the group manager, all application providers, and all users except the target user i^*. It also generates a (malicious) group public key gpk.
3. Then $\mathcal{A}$ is allowed to access the oracle $O_{List}[\mathcal{U} \cup \{GM\} - \{i^*\}]$, $O_{Join-U}[gpk]$, and $O_{Auth-U}[gpk]$ multiple times in any order.
4. Finally, the challenger checks authentication logs of applications providers. The adversary $\mathcal{A}$ succeeds in the experiment if there exists an application provider AP with public information ID_{AP}, k_{AP} and public log $\mathcal{LOG}_{AP}$ s.t. the result of running the public tracing algorithm on input $(gpk, \mathcal{LOG}_{AP}, \mathcal{LIST})$ consists i^*.

Exculpability for the group manager. A k-times anonymous authentication is exculpable for the group manager if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligible.

1. In the beginning, the challenger generates the group key pair (gpk, gsk) by running the setup procedure, and sends gpk to the adversary $\mathcal{A}$.
2. Then, the adversary $\mathcal{A}$, who controls all application providers and all users, is allowed to access the oracle $O_{List}[\mathcal{U}]$ and $O_{Join-GM}[gpk, gsk]$ multiple times in any order. Here, whenever the adversary queries the oracle $O_{Join-GM}[gpk, gsk]$ with an identity i and a user public key mpk, it should also query the oracle $O_{List}[\mathcal{U}]$ with input $(1, i, mpk)$.
3. Finally, the challenger checks authentication logs of applications providers. The adversary $\mathcal{A}$ succeeds in the experiment if there exists an application provider AP with public information ID_{AP}, k_{AP} and public log $\mathcal{LOG}_{AP}$ s.t. the result of running the public tracing algorithm on input $(gpk, \mathcal{LOG}_{AP}, \mathcal{LIST})$ consists GM.

E.3 Definition of The Blacklistable Anonymous Credentials

In this section, we recall the syntax and the security model of blacklistable anonymous credential system, which is defined in [TAKS07]. A blacklistable anonymous credential system involves three types of entities, namely, the group manager, the service providers, and the users. It also consists of three protocols (algorithms):

- **Setup.** In the setup protocol, the group manager generates its group public key and group secret key, and publish the group public key.
- **Registration.** The registration protocol is run between a user and the group manager, and after the procedure, the user, who is called a group member now, obtains a member public key and a member secret key.
- **Authentication.** The authentication protocol is run between a user and a service provider. The service provider uses a blacklist, which consists of tickets of authentication events, during the protocol. After the protocol, a ticket for this authentication event will be output[8], and the service provider accepts the user iff the user is a legally registered group member and is not related to an item in the blacklist.

In the basic version of the blacklistable anonymous credential system, a user is banned from accessing the services of a service provider once he is put into the blacklist. However, in practice, more fine-grained access control is needed. One simple but useful modification is to ban a user iff he has been put into the blacklist more than a pre-defined upper bound, say d, times, and this is called the "d-strike-out" policy.

Correctness of the blacklistable anonymous credential system requires that if all entities in the system are honest, then for any legitimately registered user and any service provider, the user is able to successfully authenticate herself to the service provider with overwhelming probability if she is not related to any item in the

[8] The ticket will be put in the blacklist in later authentications if the user misbehaves when accessing the services after being accepted by the service provider.

blacklist (or she is not related to more than d items in the blacklist if "d-strike-out" policy is used) of the service provider during the authentication.
Security of the blacklistable anonymous credential system requires that the system is blacklistable, Anonymous, and non-frameable, which can be formally defined by the following experiments.[9] In these experiments, the challenger will maintain an internal state *state*, three counters I, J, K, and six sets $\mathcal{U}_h, \mathcal{U}_g, \mathcal{U}_u, \mathcal{S}_h, \mathcal{S}_s, \mathcal{MA}$. Here, the counters I, J, K records the current number of registered users, service providers and authentication events in an experiment respectively and are initially set as 0. The sets $\mathcal{U}_h, \mathcal{U}_g, \mathcal{U}_u, \mathcal{S}_h, \mathcal{S}_s, \mathcal{MA}$ contains the current honest users, honest users registered by a malicious group manager, malicious user, honest service providers, malicious service providers, and transcripts of authentication event launched by a malicious user respectively, and are initially set as $\emptyset$. Also, The adversary is allowed to query some of the following oracles in each experiment:

- The oracle $\mathcal{REG}$. This oracle allows the adversary to register an honest user with the honest group manager. Upon invocation, the oracle simulates the registration protocol between an honest user and the honest group manager (here, we allow the adversary to view the executation of the protocol, i.e. it can get the protocol transcript). Then it increments I by 1, sets $state = state \| (I, \varpi_I, \varrho_I)$, and adds I to $\mathcal{U}_h$, where ϖ_I is the protocol transcript and ϱ_I is the internal state of the honest user when participating the protocol. The newly registered user is indexed by I.
- The oracle $\mathcal{REG}_u$. This oracle allows the adversary to register a corrupt user with the honest group manager. Upon invocation, the oracle plays the role of the group manager and interacts with the adversary in the registration protocol. Then it increments I by 1, sets $state = state \| (I, \varpi_I, \perp)$, and adds I to $\mathcal{U}_u$, where ϖ_I is the protocol transcript. The newly registered user is indexed by I.
- The oracle $\mathcal{REG}_g$. This oracle allows the adversary to register an honest user with the corrupt group manager. Upon invocation, the oracle plays the role of a user and interacts with the adversary in the registration protocol. Then it increments I by 1, sets $state = state \| (I, \varpi_I, \varrho_I)$, and adds I to $\mathcal{U}_g$, where ϖ_I is the protocol transcript and ϱ_I is the internal state of the honest user when participating the protocol. The newly registered user is indexed by I.
- The oracle $\mathcal{ADD}$-$\mathcal{SP}$. This oracle allows the adversary to introduce a service provider with fresh identity *sid* into the system. Upon invocation, the oracle increments J by 1 and adds J to $\mathcal{S}_h$, The service provider is indexed by J.
- The oracle $\mathcal{CORRUPT}$-$\mathcal{SP}$. This oracle allows the adversary to introduce a corrupt service provider with fresh identity *sid* into the system. Upon invocation, the oracle increments J by 1 and adds J to $\mathcal{S}_s$, The service provider is indexed by J.
- The oracle $\mathcal{AUTH}$. This oracle allows the adversary to eavesdrop an authentication run between an honest user and an honest service provider. On input (i, j) such that $i \in \mathcal{U}_h \cup \mathcal{U}_g$ and $j \in \mathcal{S}_h$, the oracle simulates the authentication

[9] For simplicity, we only define the security experiments for the basic case, and security experiments for blacklistable anonymous credential system with advanced policies, e.g. the d-strike-out policy, can be defined in a similar way.

protocol between an honest user i and an honest service provider j (here, we allow the adversary to view the executation of the protocol, i.e. it can get the protocol transcript). Then it increments K by 1, sets $state = state \| (K, \varpi_K, \varrho_K)$, where ϖ_K is the protocol transcript and ϱ_K is the internal state of the honest user when participating the protocol.

- The oracle $\mathcal{AUTH}_u$. This oracle allows a corrupt user to authenticate to an honest service provider. On input $j \in \mathcal{S}_h$, the oracle plays the role of the service provider j and interacts with the adversary in the authentication protocol. Then it increments K by 1, sets $state = state \| (K, \varpi_K, \perp)$, and adds K to $\mathcal{MA}$, where ϖ_K is the protocol transcript.
- The oracle $\mathcal{AUTH}_s$. This oracle allows the adversary to have an honest user to authenticate to a corrupt service provider. On input $i \in \mathcal{U}_h \cup \mathcal{U}_g$ and $j \in \mathcal{S}_s$, the oracle plays the role of the user i to authenticate to the service provider j and interacts with the adversary in the authentication protocol, Then it increments K by 1, sets $state = state \| (K, \varpi_K, \varrho_K)$, where ϖ_K is the protocol transcript and ϱ_K is the internal state of the honest user when participating the protocol.
- The oracle $\mathcal{ADD\text{-}BL}$. This oracle allows the adversary to affect an honest service provider to judge a user as having misbehaved during an authenticated session. On input $j \in \mathcal{S}_h$ and $k \in [1, K]$, the oracle adds the ticket τ_k, which is extracted from the protocol transcript ϖ_k of the kth authentication event, to the blacklist of the service provider j.
- The oracle $\mathcal{REMOVE\text{-}BL}$. This oracle allows the adversary to affect an honest SP to forgive a user for her misbehavior during an authenticated session. On input $j \in \mathcal{S}_h$ and a ticket τ in the blacklist of the service provider j, the oracle removes τ from that blacklist.
- The oracle $\mathcal{QUERY}$. This oracle is the challenge oracle for the anonymity experiment. On input $i_0^*, i_1^* \in \mathcal{U}_g$ and $j^* \in [1, J]$, the oracle samples $b \xleftarrow{\$} \{0, 1\}$. Then if $j \in \mathcal{S}_h$, it runs the oracle $\mathcal{AUTH}$ with input i_b^*, j^*; and if $j \in \mathcal{S}_s$, it runs the oracle $\mathcal{AUTH}_s$ with input i_b^*, j^*.

Here, the adversary is allowed to execute many registration oracles concurrently, i.e. we require the blacklistable anonymous credential systems to support concurrent enronment; but we assume that each authentication oracle query is atomic.

Blacklistability. A blacklistable anonymous credential system is blacklistable if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligible.

1. In the beginning, the challenger generates the group key pair (gpk, gsk) by running the setup procedure, and sends gpk to the adversary $\mathcal{A}$.
2. Then $\mathcal{A}$ is allowed to query all oracles defined above except the oracle $\mathcal{REG}_g$ (i.e. $\mathcal{A}$ is not allowed to corrupt the group manager) and the oracle $\mathcal{QUERY}$.

3. Finally, $\mathcal{A}$ succeeds in the game if there exists a series of ordered oracle queries (in chronological order):

$$\begin{pmatrix} k_1 \leftarrow \mathcal{AUTH}_u(j), & \mathcal{ADD\text{-}BL}(j, k_1), \\ k_2 \leftarrow \mathcal{AUTH}_u(j), & \mathcal{ADD\text{-}BL}(j, k_2), \\ & \vdots \\ k_{n-1} \leftarrow \mathcal{AUTH}_u(j), & \mathcal{ADD\text{-}BL}(j, k_{n-1}), \\ k_n \leftarrow \mathcal{AUTH}_u(j) & \end{pmatrix}$$

that

- $j \in \mathcal{S}_h$.
- All n queries are accepted by j.
- $n > \|\mathcal{U}_u\| + Q_R$, where Q_R is the number of queries made by $\mathcal{A}$ to the $\mathcal{REMOVE\text{-}BL}$ oracle.

We call this series "bad series" for short.

Anonymity. A blacklistable anonymous credential system is anonymous if for any probabilistic polynomial time admissible adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligibly close to $1/2$.

1. In the beginning, the challenger generates the group key pair (gpk, gsk) by running the setup procedure, and sends (gsk, gpk) to the adversary $\mathcal{A}$.
2. Then $\mathcal{A}$ is allowed to query all oracles defined above except the oracle $\mathcal{REG}$, the oracle $\mathcal{REG}_u$ (this is because $\mathcal{A}$ acts as the group manager and there is no honest group manager in this experiment), and the oracle $\mathcal{QUERY}$.
3. Then, $\mathcal{A}$ enters the challenge phase and queries the oracle $\mathcal{QUERY}(i_0^*, i_1^*, j^*)$, where $i_0^*, i_1^* \in \mathcal{U}_g$ and $j^* \in [1, J]$. Let k^* be the index for the authentication oracle queried implicity in the oracle $\mathcal{QUERY}$.
4. After the challenge phase, $\mathcal{A}$ is further allowed to query all oracles defined above except the oracle $\mathcal{REG}$, the oracle $\mathcal{REG}_u$, and the oracle $\mathcal{QUERY}$.
5. Finally, $\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$, where b is the bit used in the challenge oracle. Here we say an adversary $\mathcal{A}$ is admissible if either of the following two conditions holds.
   - The blacklist of service provider j^* used in the challenge oracle executation does not contain ticket extracted from an authentication protocol transcript that is generated by an authentication oracle query involving i_0^* or i_1^*; $\mathcal{A}$ does not query the oracle $AUTH_s(i_d^*, *)$ with a blacklist contains the ticket extracted from ϖ_{k^*}; $\mathcal{A}$ does not query the oracle $\mathcal{ADD\text{-}BL}(*, k^*)$.
   - The blacklist of service provider j^* used in the challenge oracle executation contains at least one ticket extracted from an authentication protocol transcript that is generated by an authentication oracle query involving i_0^* and at least one ticket extracted from an authentication protocol transcript that is generated by an authentication oracle query involving i_1^*.

Nonframeability. A blacklistable anonymous credential system is nonframeable if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligible.

1. In the beginning, the challenger generates the group key pair (gpk, gsk) by running the setup procedure, and sends (gsk, gpk) to the adversary $\mathcal{A}$.

2. Then $\mathcal{A}$ is allowed to query all oracles defined above except the oracle $\mathcal{REG}$, the oracle $\mathcal{REG}_u$ (this is because $\mathcal{A}$ acts as the group manager and there is no honest group manager in this experiment), and the oracle $\mathcal{QUERY}$.
3. Finally, $\mathcal{A}$ queries the oracle $\mathcal{AUTH}$ with $i^* \in \mathcal{U}_g$ and $j^* \in \mathcal{S}_h$, and wins in the experiment if
   - The blacklist used by j^* in this oracle query does not contain a ticket extracted from an authentication protocol transcript that is generated by an authentication oracle query involving i^*.
   - The authentication is not accepted by j^*.

F Omitted Security Proofs for Applications

Before stepping in the detailed security proofs for applications constructed in Sec.5, we first prove that weak pseudorandomness and uniqueness implies one-wayness, which is used frequently in these security proofs. More precisely, we will prove that

Lemma F.1. *For any weak pseudorandom function $G = (G.KeyGen, G.Eval)$ with uniqueness that has a super-polynomial domain and a super-polynomial range, let $k \leftarrow G.KeyGen(1^\lambda)$, x to be sampled uniformly at random from the domain of G, and $y = G_k(x)$, then for any probabilistic polynomial time adversary $\mathcal{A}$, $\Pr[k' \leftarrow \mathcal{A}^{\mathcal{O}_k()}(x, y) : G_{k'}(x) = y] \leq negl(\lambda)$, where $\mathcal{O}_k$ samples x_i uniformly from the domain of G and outputs $(x_i, G_k(x_i))$ each time being queried.*

At first glance, weak pseudorandomness itself is strong enough to imply the onewayness needed here. However, if multiple keys can be used to explain an input/output pair, it may be extreamly easy to find one. To see this, let $F = (F.KeyGen, F.Eval)$ be any weak pseudorandom function, then we define $F' = (F'.KeyGen, F'.Eval)$. Here $F'.KeyGen(1^\lambda)$ first samples $k_1 \xleftarrow{\$} \{0,1\}^\lambda$ and k_3 uniformly at random from the range of F, then computes $k_2 \leftarrow F.KeyGen(1^\lambda)$, and outputs the key $k = (k_1, k_2, k_3)$; $F'.Eval((k_1, k_2, k_3), x)$ outputs $F_{k_2}(x)$ if $k_1 \neq 0^\lambda$, and outputs k_3 otherwise. F' is also a weak pseudorandom function since for a key $k' = (k_1, k_2, k_3)$ sampled from the key space of F', $k_1 = 0^\lambda$ with only a negligible probability, i.e. F' behves like F in evaluating values with all but a negligible probability. Nonetheless, it is easy to check that for any given (x, y), an adversary $\mathcal{A}$ could win the game by merely set $k' = (0^\lambda, *, y)$ where we use $*$ to denote any string in the key space of F. So the uniqueness makes a difference.

Proof. We prove Lemma F.1 by showing that if there exists an probabilistic polynomial time adversary $\mathcal{A}$ that outputs a valid k' with a non-negligible probability, then we can construct an adversary $\mathcal{B}$ that breaks the weak pseudorandomness of G as follows.
In the beginnig, $\mathcal{B}$ asks for 2 input/output pair from its challenge oracle and gets $\{x_i, y_i\}_{i=1}^2$ back. Then it sneds (x_1, y_1) to $\mathcal{A}$. Each time $\mathcal{A}$ queries to the oracle $\mathcal{O}_k$, $\mathcal{B}$ queries its own challenge oracle and returns what it receives. Finally, $\mathcal{B}$ gets k' back. It then tests if $G_{k'}(x_i) = y_i$ for $i \in [1, 2]$ and outputs 1 if all equations hold.

Note that if the challenge oracle of $\mathcal{B}$ is computed by G, then the view of $\mathcal{A}$ is identical to that in the real experiment, thus it can output a valid k' with a non-negligible probability. By the uniqueness G, $k' = k$ with all but a negligible probability. So $\mathcal{B}$ will outputs 1 with a non-negligible probability in this case. On the other hand, if the challenge oralce of $\mathcal{B}$ is computed by a random function, then the probability that $\mathcal{B}$ outputs 1 is negligible. To see this, note that by the uniquesness, there exists at most one key k'' that satisfies $G_{k''}(x_1) = y_1$. Also, as the domain of G is super-polynomial large, x_1 and x_2 are distinct with all but a negligible probability. So y_2 can be viewed as sampled uniformly at random from the range of G, which equals to $G_{k''}(x_2)$ with only a negligible probability. That completes the proof. □

F.1 Proof of Theorem 5.1

Proof. Verification correctness comes from the correctness of the underlying building blocks directly, and the linking correctness comes from the correctness and the uniqueness of the wPRF with efficient protcol.

Unforgeability. Next, we prove the unforgeability of the linkable ring signature scheme via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.

• *Game 0.* This is the real unforgeability experiment defined in Appendix E.1.

• *Game 1.* This is identical to Game 0 except that when answering the signing oracle, the challenger computes the proof Π in the signature σ by generating a simulated one. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $| P_1 - P_0 | \leq negl(\lambda)$.

• *Game 2.* This is identical to Game 1 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally attempts to extract the witness $(\boldsymbol{s}^*, \boldsymbol{y}^*, \boldsymbol{y'}^*, \mathfrak{w}^*)$ from Π^* in the submitted signature σ^*, and outputs 0 (indicating that $\mathcal{A}$ fails) if it fails to extract a valid witness, where we say a witness is valid if it satisfies the statement defiend in the signing algorithm in the construction. Note that in our construction, the hash of the message $\mathfrak{m}$ (together with some other terms) will be part of the proof Π in the signature σ, thus, with all but a negligible probability, different $\mathfrak{m}$ leads to different Π. Also, since $(\mathfrak{m}^*, \mathcal{R}^*, \sigma^*)$ has not been queried to the signing oracle, $(\mathcal{R}^*, t^*, \Pi^*)$, in which $(\mathcal{R}^*, t^*)$ is part of the statement for the proof Π^*, has not appeared with all but a negligible probability. So, by the simulation extractability of the udnerlying ZKAoK systems, the challenger can succeed in extracting a valid witness with all but a negligible probability. Thus, we have $| P_2 - P_1 | \leq negl(\lambda)$.

• *Game 3.* This is identical to Game 2 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally checks whether $F_{\boldsymbol{s}^*}(\boldsymbol{A}) \in \mathcal{R}^*$, and outputs 0 if not. Indistinguishability between Game 2 and Game 3 comes from the security of the accumulator scheme and the SIS assumption directly. So, we have $| P_3 - P_2 | \leq negl(\lambda)$.

• *Game 4.* This is identical to Game 3 except that the challenger samples an integer $i^* \xleftarrow{\$} [1, n]$ in the beginning and outputs 0 if $F_{\boldsymbol{s}^*}(\boldsymbol{A}) \neq pk_{i^*}$. As the integer i^* is independent of the view of $\mathcal{A}$, and the set $\mathcal{R}^* \subseteq \mathcal{S}$, so the probability that $F_{\boldsymbol{s}^*}(\boldsymbol{A}) =$

pk_{i^*} is $1/n$. Note that if the challenger succeeds in guessing i^* (namely, $F_{s^*}(\boldsymbol{A}) = pk_{i^*}$), Game 3 and Game 4 are identical, so we have $P_4 = P_3/n$, i.e. $| nP_4 - P_0 | \leq negl(\lambda)$.

It remains to show that P_4 is negligible, and this comes from Lemma F.1. To argue this, we show that if P_4 is non-negligible, then we can construct an adversary $\mathcal{B}$ that can generates a valid k' explaining the given input/output pair. More precisely, the adversary first obtains a challenge pair $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{y}})$. Then it queries its oracle and obtains another pair $(\hat{\boldsymbol{B}}, \hat{\boldsymbol{t}})$ and samples an integer $i^* \xleftarrow{\$} [1, n]$. Next, it simulates the environment for $\mathcal{A}$. On the first phase, $\mathcal{B}$ generates the public parameter and n public keys honestly, except that it sets the matrix $\boldsymbol{A}, \boldsymbol{B}$ in the public parameter to be $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$ respectively and that it sets $pk_{i^*} = \hat{\boldsymbol{y}}$ (and set sk_{i^*} as empty). Then in the second phase, when answering a signing oracle query $(i^*, *, *)$, $\mathcal{B}$ sets the tag $\boldsymbol{t} = \hat{\boldsymbol{t}}$ and simulate the proof Π; and if i^* is queried to the corrupt oracle, $\mathcal{B}$ aborts the simulation and reutrns a random value to its challenger. Finally, when $\mathcal{A}$ returns the challenge $(\mathfrak{m}^*, \mathcal{R}^*, \sigma^*)$, $\mathcal{B}$ extracts the witness $(\boldsymbol{s}^*, \boldsymbol{y}^*, \boldsymbol{y'}^*, \mathfrak{w}^*)$ from Π^* in the submitted signature σ^*, and returns $\boldsymbol{s}^*$ back. Note that $\mathcal{B}$ can perfectly simulate Game 4 unless i^* is queried to the corrupt oracle, and $\mathcal{A}$ will definitely fail in Game 4 if i^* is queried to the corrupt oracle, so with probability P_4, $F_{s^*}(\boldsymbol{A}) = pk_{i^*} = \hat{\boldsymbol{y}}$, i.e. $\mathcal{B}$ returns the valid key for $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{y}})$. That completes the proof for unforgeability.

Linkable-Anonymity. Next, we prove the linkable-anonymity of the linkable ring signature scheme via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.

• *Game 0.* This is the real linkable-anonymity experiment defined in Appendix E.1.

• *Game 1.* This is identical to Game 0 except that when answering the challenge query in Phase 3, the challenger computes the proof Π^* in the signature σ^* by generating a simulated one. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $| P_1 - P_0 | \leq negl(\lambda)$. It is worth noting that in Game 1, σ^* can be generated with merely the knowledge of the public parameter, the ring $\mathcal{R}^*$ and the tag t^*.

• *Game 2.* This is identical to Game 1 except that the challenger samples two integers $j_0^*, j_1^* \xleftarrow{\$} [1, n]$ in the beginning, then in Phase 2, it aborts and outputs a random bit (indicating whether $\mathcal{A}$ succeed) if j_0^* or j_1^* is queried to the signing oracle or the corrupt oracle. The challenger also aborts and outputs a random bit if $i_0^* \neq j_0^*$ or $i_1^* \neq j_1^*$ in Phase 3. As the integers j_0^* and j_1^* are independent of the view of $\mathcal{A}$ untill the challenger abots, we have $P_2 = 1/n^2 \cdot P_1 + (1 - 1/n^2) \cdot 1/2 = 1/2 + (P_1 - 1/2)/n^2$.

• *Game 3.* This is identical to Game 2 except that $pk_{j_0^*}$ is sampled uniformly at random from the range of F ($sk_{j_0^*}$ is empty) in Phase 1, and t^* is also sampled uniformly at random from the range of F if $b = 0$. Indistinguishability between Game 2 and Game 3 comes from the weak pseudorandomness of the underlying wPRF with efficient protocol directly. So, we have $| P_3 - P_2 | \leq negl(\lambda)$.

• *Game 4.* This is identical to Game 3 except that $pk_{j_1^*}$ is sampled uniformly at random from the range of F ($sk_{j_1^*}$ is empty) in Phase 1, and t^* is also sampled uniformly at random from the range of F if $b = 1$. Indistinguishability between Game 3 and Game 4 comes from the weak pseudorandomness of the underlying

wPRF with efficient protocol directly. So, we have $| P_4 - P_3 | \leq negl(\lambda)$. Note that in Game 4, the view of $\mathcal{A}$ is independent of the challenge bit b, so we have $P_4 = 1/2$, i.e. $| P_0 - 1/2 | \leq negl(\lambda)$. That completes the proof of linkable-anonymity.

Linkability. Next, we prove the linkability of the linkable ring signature scheme via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.

• *Game 0.* This is the real linkability experiment defined in Appendix E.1.

• *Game 1.* This is identical to Game 0 except that when answering the signing oracle, the challenger computes the proof Π in the signature σ by generating a simulated one. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $| P_1 - P_0 | \leq negl(\lambda)$.

• *Game 2.* This is identical to Game 1 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally attempts to extract the witness $(\boldsymbol{s}_1^*, \boldsymbol{y}_1^*, \boldsymbol{y}_1'^*, \mathfrak{w}_1^*)$ from Π_1^* in the submitted signature σ_1^* and the witness $(\boldsymbol{s}_2^*, \boldsymbol{y}_2^*, \boldsymbol{y}_2'^*, \mathfrak{w}_2^*)$ from Π_2^* in the submitted signature σ_2^*, and outputs 0 (indicating that $\mathcal{A}$ fails) if it fails to extract either valid witness, where we say a witness is valid if it satisfies the statement defiend in the signing algorithm in the construction. Note that in our construction, the hash of the message $\mathfrak{m}$ (together with some other terms) will be part of the proof Π in the signature σ, thus, with all but a negligible probability, different $\mathfrak{m}$ leads to different Π. Also, since both $(\mathfrak{m}_1^*, \mathcal{R}_1^*, \sigma_1^*)$ and $(\mathfrak{m}_2^*, \mathcal{R}_2^*, \sigma_2^*)$ have not been queried to the signing oracle, with all but a negligible probability, neither $(\mathcal{R}_1^*, t_1^*, \Pi_1^*)$, in which $(\mathcal{R}_1^*, t_1^*)$ is part of the statement for the proof Π_1^*, nor $(\mathcal{R}_2^*, t_2^*, \Pi_2^*)$, in which $(\mathcal{R}_2^*, t_2^*)$ is part of the statement for the proof Π_2^*, has appeared. So, by the simulation extractability of the udnerlying ZKAoK systems, the challenger can succeed in extracting valid witnesses with all but a negligible probability. Thus, we have $| P_2 - P_1 | \leq negl(\lambda)$.

• *Game 3.* This is identical to Game 2 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally checks whether $F_{\boldsymbol{s}_1^*}(\boldsymbol{A}) \in \mathcal{R}_1^* \wedge F_{\boldsymbol{s}_2^*}(\boldsymbol{A}) \in \mathcal{R}_2^*$, and outputs 0 if this is not the case. Indistinguishability between Game 2 and Game 3 comes from the security of the accumulator scheme and the SIS assumption directly. So, we have $| P_3 - P_2 | \leq negl(\lambda)$.

• *Game 4.* This is identical to Game 3 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally checks whether $\boldsymbol{s}_1^* = \boldsymbol{s}_2^*$, and outputs 0 if this is the case. Game 3 and Game 4 are identical unless $\mathcal{A}$ wins in Game 3 but $\boldsymbol{s}_1^* = \boldsymbol{s}_2^*$. This cannot occur due to the correctness of the wPRF with efficient protocol. So, we have $| P_4 - P_3 | = 0$. Note that if $\mathcal{A}$ wins in Game 4, $\boldsymbol{s}_1^* \neq \boldsymbol{s}_2^*$, so by the uniqueness of wPRF with efficient protocol, $F_{\boldsymbol{s}_1^*}(\boldsymbol{A}) \neq F_{\boldsymbol{s}_2^*}(\boldsymbol{A})$. Also, if $\mathcal{A}$ wins, all but at most one keys in $\mathcal{R}_1^* \cup \mathcal{R}_2^*$ are in $\mathcal{S}\backslash\mathcal{CO}$, so at least for one $d \in [1, 2]$, $F_{\boldsymbol{s}_d^*}(\boldsymbol{A})$ is in $\mathcal{S}\backslash\mathcal{CO}$, and we denote this $\boldsymbol{s}_d^*$ as $\hat{\boldsymbol{s}}^*$.

• *Game 5.* This is identical to Game 4 except that the challenger samples an integer $i^* \xleftarrow{\$} [1, n]$ in the beginning and outputs 0 if $F_{\hat{\boldsymbol{s}}^*}(\boldsymbol{A}) \neq pk_{i^*}$. As the integer i^* is independent of the view of $\mathcal{A}$, so the probability that $F_{\hat{\boldsymbol{s}}^*}(\boldsymbol{A}) = pk_{i^*}$ is $1/n$. Note that if the challenger succeeds in guessing i^* (namely, $F_{\hat{\boldsymbol{s}}^*}(\boldsymbol{A}) = pk_{i^*}$), Game 4 and Game 5 are identical, so we have $P_5 = P_4/n$, i.e. $| nP_5 - P_0 | \leq negl(\lambda)$.

Finally, we can prove that P_5 is negligible by using Lemma F.1, just as we have done in proving the unforgeability of the linkable ring signature scheme. That completes the proof for linkablity.

Non-Slanderability. Next, we prove the non-slanderability of the linkable ring signature scheme via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.

• *Game 0.* This is the real non-slanderability experiment defined in Appendix E.1.

• *Game 1.* This is identical to Game 0 except that the challenger samples an integer $i^* \xleftarrow{\$} [1, n]$ in the beginning and outputs 0 if $i^* \neq \hat{j}$. As the integer i^* is independent of the view of $\mathcal{A}$, the probability that $i^* = \hat{j}$ is $1/n$. Thus, we have $P_1 = P_0/n$.

• *Game 2.* This is identical to Game 1 except that when answering the signing oracle with query $(i^*, \cdot, \cdot)$, the challenger computes the proof Π in the signature σ by generating a simulated one. Indistinguishability between Game 1 and Game 2 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $| P_2 - P_1 | \leq negl(\lambda)$.

• *Game 3.* This is identical to Game 2 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally attempts to extract the witness $(\boldsymbol{s}^*, \boldsymbol{y}^*, \boldsymbol{y'}^*, \mathfrak{w}^*)$ from Π^* in the submitted signature σ^*, and outputs 0 (indicating that $\mathcal{A}$ fails) if it fails to extract the valid witness, where we say a witness is valid if it satisfies the statement defiend in the signing algorithm in the construction. Note that in our construction, the hash of the message $\mathfrak{m}$ (together with some other terms) will be part of the proof Π in the signature σ, thus, with all but a negligible probability, different $\mathfrak{m}$ leads to different Π. Also, if $\mathcal{A}$ wins in Game 3, $(i^*, \mathfrak{m}^*, \mathcal{R}^*, \sigma^*)$ has not been queried to the signing oracle, so, with all but a negligible probability, $(\mathcal{R}^*, t^*, \Pi^*)$, in which $(\mathcal{R}^*, t^*)$ is part of the statement for the proof Π^*, is not a copy of a simulated proof made by the challenger. So, by the simulation extractability of the udnerlying ZKAoK systems, the challenger can succeed in extracting valid witnesses with all but a negligible probability. Thus, we have $| P_3 - P_2 | \leq negl(\lambda)$.

Now, in Game 3, we have $F_{s^*}(\boldsymbol{B}) = \hat{\boldsymbol{t}}$ if $\mathcal{A}$ wins, where $\hat{\boldsymbol{t}}$ is the tag for $\hat{\sigma}$. So by Lemma F.1, P_3 is negligible. We reamrk that when making the reduction, we will set the challenge pair of $\mathcal{B}$ as $(\boldsymbol{B}, \hat{\boldsymbol{t}})$, which is slightly different from what has done when proving the unforgeability and the linkability. That completes the proof for non-slanderability. □

F.2 Proof of Theorem 5.2

Proof. Correctness comes from the correctness of the underlying building blocks directly.

Anonymity. Next, we prove the anonymity of the k-times anonymous authentication protocol via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.

• *Game 0.* This is the real anonymity experiment defined in Appendix E.2.

• *Game 1.* This is identical to Game 0 except that when the challenger is required to authenticate on behalf of i_0 or i_1, it generates a simulated proof Π. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $| P_1 - P_0 | \leq negl(\lambda)$. It is worth noting that in Game 1, the response $(\boldsymbol{t}, \check{\boldsymbol{t}}, \Pi)$ in an authentication event involving i_0 or i_1 can be generated with merely the knowledge of the public parameter, the public information and the challenge of the counterpart application provider and the tag $(\boldsymbol{t}, \check{\boldsymbol{t}})$.
• *Game 2.* This is identical to Game 1 except that $\boldsymbol{y}_{i_0}$, which is part of the public key of i_0, is sampled uniformly from the range of F, and the challenger samples $(\boldsymbol{t}, \check{\boldsymbol{t}})$ uniformly from the range of F when it is required to authenticate on behalf of i_0. Since during the whole procedure of Game 1 and Game 2, the challenger is only required to compute F on random inputs and it will not be required to compute F on the same input twice, indistinguishability between Game 1 and Game 2 comes from the weak pseudorandomness of the underlying wPRF with efficient protocol directly. So, we have $| P_2 - P_1 | \leq negl(\lambda)$.
• *Game 3.* This is identical to Game 2 except that $\boldsymbol{y}_{i_1}$, which is part of the public key of i_1, is sampled uniformly from the range of F, and the challenger samples $(\boldsymbol{t}, \check{\boldsymbol{t}})$ uniformly from the range of F when it is required to authenticate on behalf of i_1. Similarly, indistinguishability between Game 2 and Game 3 comes from the weak pseudorandomness of the underlying wPRF with efficient protocol directly. So, we have $| P_3 - P_2 | \leq negl(\lambda)$.
Now, in Game 3, as the adversary can only access the oracle O_{Query} one time of the form $(d, *)$ for each queried $d \in \{0, 1\}$, and can only access the oracle $O_{Auth-U}[gpk]$ $k - 1$ times of the form $(i, (ID^*, M^*))$ for each queried $i \in \{i_0, i_1\}$, the challenger is only required to authenticate with the application provider with public information (ID^*, k^*) up to k^* times on behalf of i_0 and i_1 respectively. Thus, the query oracle will work normally for $d \in \{0, 1\}$. Therefore, the executations of the query oracle are independent of the bit b, i.e. the view of $\mathcal{A}$ is independent of b in Game 3 and $P_3 = 1/2$. So, we have $| P_0 - 1/2 | \leq negl(\lambda)$ and that completes the proof of anonymity.

Detectability. Next, we prove the detectability of the k-times anonymous authentication protocol via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.
• *Game 0.* This is the real detectability experiment defined in Appendix E.2.
• *Game 1.* This is identical to Game 0 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally attempts to extract the witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma)$ from each proof Π in the authentication log, and outputs 0 (indicating that $\mathcal{A}$ fails) if it fails to extract a valid witness for some proof Π in the authentication log, where we say a witness is valid if it satisfies the statement defiend in the authentication protocol in the construction. Indistinguishability between Game 0 and Game 1 comes from the extractability property of the underlying ZKAoK systems directly. Thus, we have $| P_1 - P_0 | \leq negl(\lambda)$.
• *Game 2.* This is identical to Game 1 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma)$, $\boldsymbol{y}$

has not been queried to the $O_{Join-GM}[gpk, gsk]$ oracle. Indistinguishability between Game 1 and Game 2 comes from the unforgeability of the underlying CL signature scheme directly. Thus, we have $| P_2 - P_1 | \leq negl(\lambda)$.
• *Game 3.* This is identical to Game 2 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for two extracted witness $(\boldsymbol{s}_1, \boldsymbol{y}_1, \boldsymbol{B}_1, \check{\boldsymbol{B}}_1, \check{\boldsymbol{t}'}_1, \boldsymbol{b'}_1, \mathfrak{w}_1, \sigma_1)$, and $(\boldsymbol{s}_2, \boldsymbol{y}_2, \boldsymbol{B}_2, \check{\boldsymbol{B}}_2, \check{\boldsymbol{t}'}_2, \boldsymbol{b'}_2, \mathfrak{w}_2, \sigma_2)$, $\boldsymbol{y}_1 = \boldsymbol{y}_2$ but $\boldsymbol{s}_1 \neq \boldsymbol{s}_2$. Indistinguishability between Game 2 and Game 3 comes from the uniqueness of the underlying wPRF with efficient protocols directly. Thus, we have $| P_3 - P_2 | \leq negl(\lambda)$. Note that in Game 3, if the adversary succeeds, it only use at most $\|\mathcal{LIST}\|$ different secret keys.
• *Game 4.* This is identical to Game 3 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}'}, \boldsymbol{b'}, \mathfrak{w}, \sigma)$ from $\mathcal{LOG}_{AP}$, $(\boldsymbol{B}, \check{\boldsymbol{B}})$ is not in the tag bases set $\mathcal{B}_{AP}$ of the application provider AP. Indistinguishability between Game 3 and Game 4 comes from the security of the accumulator scheme and the SIS assumption directly. Thus, we have $| P_4 - P_3 | \leq negl(\lambda)$.
Now, in Game 4, if the adversary succeeds, for each $\mathcal{LOG}_{AP}$ it can only use at most k_{AP} different tag bases, and at most $\|\mathcal{LIST}\|$ different secret keys, thus it can only generates tags $(\boldsymbol{t}, \check{\boldsymbol{t}})$ with at most $k_{AP} \cdot \|\mathcal{LIST}\|$ different $\boldsymbol{t}$. Therefore, it cannot succeeds in Game 4, i.e. $P_4 = 0$. That completes the proof for detectability.

Before stepping into the detailed proofs of exculpabilities of the k-times anonymous authentication protocol, we first prove a useful lemma.

Lemma F.2. *Let* F *be a secure wPRF with efficient protocols with strong uniqueness,* l *be any polynomial,* $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_l$ *are sampled uniformly at random from the domain of* F*. Then the probability that there exists secret keys* $\boldsymbol{s}_1, \boldsymbol{s}_2$ *of* F *and* i, $j \in [1, l]$ *that* $\boldsymbol{s}_1 \neq \boldsymbol{s}_2$ *but* $\mathsf{F}_{s_1}(\boldsymbol{A}_i) = \mathsf{F}_{s_2}(\boldsymbol{A}_j)$ *is negligible. Note that here we donot require that* $i \neq j$*.*

Proof. For each $(i, j) \in [1, l] \times [1, l]$, if $i = j$ then $\Pr[\exists \boldsymbol{s}_1 \neq \boldsymbol{s}_2 : \mathsf{F}_{s_1}(\boldsymbol{A}_i) = \mathsf{F}_{s_2}(\boldsymbol{A}_j)] \leq negl(\cdot)$ due to the uniqueness of F; otherwise, $\Pr[\exists \boldsymbol{s}_1 \neq \boldsymbol{s}_2 : \mathsf{F}_{s_1}(\boldsymbol{A}_i) = \mathsf{F}_{s_2}(\boldsymbol{A}_j)] \leq negl(\cdot)$ due to the strong uniqueness of F. Thus, by the union bound, $\Pr[\exists i, j, \boldsymbol{s}_1 \neq \boldsymbol{s}_2 : \mathsf{F}_{s_1}(\boldsymbol{A}_i) = \mathsf{F}_{s_2}(\boldsymbol{A}_j)] \leq negl(\cdot)$. That completes the proof. □

Exculpability for users. Next, we prove the exculpability for users of the k-times anonymous authentication protocol via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i. For simplicity of exposition, we call a pair of items $((\boldsymbol{t}_1, \check{\boldsymbol{t}}_1, c_1, \mathfrak{m}_1, \Pi_1), (\boldsymbol{t}_2, \check{\boldsymbol{t}}_2, c_2, \mathfrak{m}_2, \Pi_2))$ in an authentication log "bad pairs" if they lead to an accusation of the user i^*, namely $\boldsymbol{t}_1 = \boldsymbol{t}_2$, $c_1 \neq c_2$ and $(c_1 - c_2)^{-1} \cdot (\check{\boldsymbol{t}}_1 - \check{\boldsymbol{t}}_2)$ equals to the public key y^* of i^*. Also, we call an item in an authentication log "bad item" if it belongs to a "bad pair" and is not generated by i^* itself (i.e. the challenger).
• *Game 0.* This is the real exculpability for users experiment defined in Appendix E.2.
• *Game 1.* This is identical to Game 0 except that when the challenger is required to authenticate on behalf of i^*, it generates a simulated proof Π. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the

underlying ZKAoK systems directly. So, we have $| P_1 - P_0 | \leq negl(\lambda)$. It is worth noting that in Game 1, the response $(\boldsymbol{t}, \check{\boldsymbol{t}}, \Pi)$ in an authentication event involving i^* can be generated with merely the knowledge of the public parameter, the public information and the challenge of the counterpart application provider and the tag $(\boldsymbol{t}, \check{\boldsymbol{t}})$.

• *Game 2.* This is identical to Game 1 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally attempts to extract the witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma)$ from each proof Π in a "bad item", and outputs 0 (indicating that $\mathcal{A}$ fails) if it fails to extract a valid witness for some proof Π in a "bad item", where we say a witness is valid if it satisfies the statement defiend in the authentication protocol in the construction. Indistinguishability between Game 1 and Game 2 comes from the simulation extractability property of the underlying ZKAoK systems directly, since each "bad item" contains a fresh statement/proof pair. Thus, we have $| P_2 - P_1 | \leq negl(\lambda)$.

• *Game 3.* This is identical to Game 2 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma)$ from $\mathcal{LOG}_{AP}$, $(\boldsymbol{B}, \check{\boldsymbol{B}})$ is not in the tag bases set $\mathcal{B}_{AP}$ of the application provider AP. Indistinguishability between Game 2 and Game 3 comes from the security of the accumulator scheme and the SIS assumption directly. Thus, we have $| P_3 - P_2 | \leq negl(\lambda)$.

• *Game 4.* This is identical to Game 3 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if there exist two extracted witness $(\boldsymbol{s}_1, \boldsymbol{y}_1, \boldsymbol{B}_1, \check{\boldsymbol{B}}_1, \check{\boldsymbol{t}}'_1, \boldsymbol{b}'_1, \mathfrak{w}_1, \sigma_1)$, and $(\boldsymbol{s}_2, \boldsymbol{y}_2, \boldsymbol{B}_2, \check{\boldsymbol{B}}_2, \check{\boldsymbol{t}}'_2, \boldsymbol{b}'_2, \mathfrak{w}_2, \sigma_2)$, that are extracted from two "bad items" formating a "bad pair" and $\boldsymbol{s}_1 \neq \boldsymbol{s}_2$. Indistinguishability between Game 3 and Game 4 comes from Lemma F.2 directly. Thus, we have $| P_4 - P_3 | \leq negl(\lambda)$. Note that in Game 4, if $\mathcal{A}$ succeeds, for each "bad pair" consisting of two "bad items", the extracted secret keys $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are identical, thus, we have $\check{\boldsymbol{t}}'_1 = \check{\boldsymbol{t}}'_2$ and $\boldsymbol{y}_1 = \boldsymbol{y}_2$, which implies that $\boldsymbol{y}_1 = \boldsymbol{y}_2 = \boldsymbol{y}^*$. [10] As a result, the challenger can extract the secret key for $\boldsymbol{y}^*$ in this case.

• *Game 5.* This is identical to Game 4 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if there exists "bad pair" consisting of two "bad items". Indistinguishability between Game 4 and Game 5 comes from Lemma F.1 directly. Thus, we have $| P_5 - P_4 | \leq negl(\lambda)$.

• *Game 6.* This is identical to Game 5 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if there exist an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma)$ that is extracted from a "bad items" in a "bad pair" consisting of this "bad item" and an item generated by the challenger and $\boldsymbol{s} \neq \boldsymbol{s}^*$, where $\boldsymbol{s}^*$ is the secret key for i^*. Indistinguishability between Game 5 and Game 6 comes from Lemma F.2 directly. Thus, we have $| P_6 - P_5 | \leq negl(\lambda)$.

• *Game 7.* This is identical to Game 6 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if there exists "bad pair" consisting of a "bad item"

[10] This is because our wPRF with efficient protocol satisfies a stronger version of strong uniqueness, i.e. for any secret key not equal to $\mathbf{0}$, different inputs will be evaluated to different outputs. So, if $s_1 \neq \mathbf{0}$, we have $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are also identical, which implies that $\check{\boldsymbol{B}}_1$ and $\check{\boldsymbol{B}}_2$ are also identical; if $s_1 = \mathbf{0}$, we have $\check{\boldsymbol{t}}'_1 = \check{\boldsymbol{t}}'_2 = \boldsymbol{y}_1 = \boldsymbol{y}_2 = \mathbf{0}$.

and an item generated by the challenger. Indistinguishability between Game 6 and Game 7 comes from Lemma F.1 directly. Thus, we have $| P_7 - P_6 | \leq negl(\lambda)$.
Now in Game 7, if $\mathcal{A}$ succeeds, there must exists "bad pair" consisting of two items generated by the challenger. However, this can occur with only a negligible probability since 1) the user i^* is honest and will not attempt to authenticate more than k_{AP} times for each application provider AP and 2) all tag bases are sampled uniformly and the probability that repeated tag bases occur is negligible and 3) the probability that for the same secret key (generated by the key generation algorithm), different uniform inputs are evaluated to the same output is negligible (otherwise, the function can be distinguished from random function easily). Note that the "copy attack", where the adversary copies an item generated by i^* in the authentication log, does not work since the copied item has the same c with the original item and cannot form a "bad pair". That completes the proof for exculpability for users.

Exculpability for the group manager. Next, we prove the exculpability for the group manager of the k-times anonymous authentication protocol via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i. For simplicity of exposition, we call a pair of items $((\boldsymbol{t}_1, \check{\boldsymbol{t}}_1, c_1, \mathfrak{m}_1, \Pi_1), (\boldsymbol{t}_2, \check{\boldsymbol{t}}_2, c_2, \mathfrak{m}_2, \Pi_2))$ in an authentication log "bad pairs" if they lead to an accusation of the group manager, namely $\boldsymbol{t}_1 = \boldsymbol{t}_2, c_1 \neq c_2$ and $(c_1 - c_2)^{-1} \cdot (\check{\boldsymbol{t}}_1 - \check{\boldsymbol{t}}_2)$ has not been queried to the group manager in a join protocol. Also, we call an item in an authentication log "bad item" if it belongs to a "bad pair".
• *Game 0.* This is the real exculpability for the group manager experiment defined in Appendix E.2.
• *Game 1.* This is identical to Game 0 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally attempts to extract the witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma)$ from each proof Π in a "bad item", and outputs 0 (indicating that $\mathcal{A}$ fails) if it fails to extract a valid witness for some proof Π in a "bad item", where we say a witness is valid if it satisfies the statement defiend in the authentication protocol in the construction. Indistinguishability between Game 0 and Game 1 comes from the extractability property of the underlying ZKAoK systems directly. Thus, we have $| P_1 - P_0 | \leq negl(\lambda)$.
• *Game 2.* This is identical to Game 1 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \boldsymbol{B}, \check{\boldsymbol{B}}, \check{\boldsymbol{t}}', \boldsymbol{b}', \mathfrak{w}, \sigma)$ from $\mathcal{LOG}_{AP}$, $(\boldsymbol{B}, \check{\boldsymbol{B}})$ is not in the tag bases set $\mathcal{B}_{AP}$ of the application provider AP. Indistinguishability between Game 1 and Game 2 comes from the security of the accumulator scheme and the SIS assumption directly. Thus, we have $| P_2 - P_1 | \leq negl(\lambda)$.
• *Game 3.* This is identical to Game 2 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if there exist two extracted witness $(\boldsymbol{s}_1, \boldsymbol{y}_1, \boldsymbol{B}_1, \check{\boldsymbol{B}}_1, \check{\boldsymbol{t}}'_1, \boldsymbol{b}'_1, \mathfrak{w}_1, \sigma_1)$, and $(\boldsymbol{s}_2, \boldsymbol{y}_2, \boldsymbol{B}_2, \check{\boldsymbol{B}}_2, \check{\boldsymbol{t}}'_2, \boldsymbol{b}'_2, \mathfrak{w}_2, \sigma_2)$, that are extracted from two "bad items" formating a "bad pair" and $\boldsymbol{s}_1 \neq \boldsymbol{s}_2$. Indistinguishability between Game 2 and Game 3 comes from Lemma F.2 directly. Thus, we have $| P_3 - P_2 | \leq negl(\lambda)$.
Note that in Game 3, if $\mathcal{A}$ succeeds, for each "bad pair", the extracted secret keys $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are identical, thus, we have $\check{\boldsymbol{t}}'_1 = \check{\boldsymbol{t}}'_2$ and $\boldsymbol{y}_1 = \boldsymbol{y}_2$, which implies that $\boldsymbol{y}_1 =$

$\boldsymbol{y}_2 = \boldsymbol{y}'$, where $\boldsymbol{y}' = (c_1 - c_2)^{-1} \cdot (\check{\boldsymbol{t}}_1 - \check{\boldsymbol{t}}_2)$. [11] Recall that $\boldsymbol{y}'$ has not been queried to the group manager in a join protocol, thus, by the unforgeability of the underlying CL signature scheme, A cannot succeeds in Game 3 with a non-negligible probability. That completes the proof of exculpability for the group manager.

□

F.3 Proof of Theorem 5.3

Proof. Here, we only present security proofs for the basic version of the blacklistable anonymous credential system, and one can adapt the basic proof provided here to security proofs for blacklistable anonymous credential systems with fine-grained policies easily.
Correctness comes from the correctness of the underlying building blocks directly.

Blacklistability. Next, we prove the blacklistability of the blacklistable anonymous credential system via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.
• *Game 0.* This is the real blacklistability experiment defined in Appendix E.3.
• *Game 1.* This is identical to Game 0 except that when answering an authentication oracle on behalf of an honest user (i.e. when answering the oracle $\mathcal{AUTH}$ or the oracle $\mathcal{AUTH}_s$), the challenger generates a simulated proof Π. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $| P_1 - P_0 | \leq negl(\lambda)$. It is worth noting that in Game 1, the response $(\mu, \boldsymbol{t}, \Pi)$ in an authentication event involving an honest user can be generated with merely the knowledge of the public parameter, the blacklist of the counterpart service provider and the ticket $(\mu, \boldsymbol{t})$.
• *Game 2.* This is identical to Game 1 except that when answering an authentication oracle on behalf of an honest user (i.e. when answering the oracle $\mathcal{AUTH}$ or the oracle $\mathcal{AUTH}_s$), the challenger checks whether the user is blocked by the received blacklist via checking whether there exists a ticket generated by this user in the blacklist.
Note that in Game 2, the output of an authentication oracle involving an honest user is determined by the ticket generated this time (if have), the public information, and the history behaviors of the invloved honest user. Also, the output of a registeration oracle involving an honest user is determined by the public key of the user. Therefore, the view of $\mathcal{A}$ in Game 2 is in fact independent of the real value of these secret keys of honest users, and only depends on the functionalities of these secret keys.
Game 1 and Game 2 are identical unless $\mathcal{A}$ can put a "bad ticket" $(\mu^*, \boldsymbol{t}^*)$ into a blacklist (either via a malicious service provider or via the $\mathcal{ADD}$-$\mathcal{BL}$ oracle), where we say a ticket is bad if 1) $\mathsf{F}_{\boldsymbol{s}}(\mathcal{H}(\mu^*)) = \boldsymbol{t}^*$ for a secret key $\boldsymbol{s}$ of an honest user and 2) $(\mu^*, \boldsymbol{t}^*)$ is not generated by that honest user. Let $\mathfrak{p}$ be the probability

[11] This is because our wPRF with efficient protocol satisfies a stronger version of strong uniqueness, i.e. for any secret key not equal to $\mathbf{0}$, different inputs will be evaluated to different outputs. So, if $s_1 \neq \mathbf{0}$, we have $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are also identical, which implies that $\check{\boldsymbol{B}}_1$ and $\check{\boldsymbol{B}}_2$ are also identical; if $s_1 = \mathbf{0}$, we have $\check{\boldsymbol{t}}'_1 = \check{\boldsymbol{t}}'_2 = \boldsymbol{y}_1 = \boldsymbol{y}_2 = \mathbf{0}$.

that a bad ticket occurs in Game 2. To argue that $\mathfrak{p}$ is negligible, we show that if $\mathfrak{p}$ is non-negligible, then we can construct an adversary $\mathcal{B}$ that can break the weak pseudorandomness of the underlying wPRF with efficient protocols.

Assuming $\mathcal{A}$ makes up to Q random oracle queries, there are Q' honest users, and w.l.o.g., assume that $\mathcal{A}$ will not query the same input twice to the random oracle and will query μ to the random oracle each time generating a new ticket $(\mu, \boldsymbol{t})$. In the beginning, $\mathcal{B}$ samples $i \xleftarrow{\$} [1, Q]$ and $j \xleftarrow{\$} [1, Q']$. Then it simulates the environment for $\mathcal{A}$ identically to that in Game 2 except that it answers all oracle queries involving the jth honest user by querying its challenge oracle (this works since the view of $\mathcal{A}$ depends on the functionalities rather than the real values of secret keys of honest users, and $\mathcal{B}$ can control the generation of $\boldsymbol{A}$ and the output of $\mathcal{H}$). When the ith random oracle is queried on a value $\hat{\mu}$, $\mathcal{B}$ first queries its challenge oracle and gets a tuple $(\hat{\boldsymbol{B}}, \hat{\boldsymbol{t}})$ back. Then it sets $\mathcal{H}(\hat{\mu}) = \hat{\boldsymbol{B}}$. $\mathcal{B}$ outputs 1 if it finds $(\hat{\mu}, \hat{\boldsymbol{t}})$ in a blacklist and it is a bad ticket. Otherwise, it outputs a random bit $b \xleftarrow{\$} \{0, 1\}$. Note that if the challenge oracle of $\mathcal{B}$ returns outputs of a pseudorandom function, then $\mathcal{B}$ can perfectly simulate the view of $\mathcal{A}$, and with a non-negligible probability $\mathfrak{p}$, $\mathcal{A}$ will output a bad ticket for an honest user. Then, with probability $1/(QQ')$, $\mathcal{B}$ can succeeds in guessing that honest user and locates the bad ticket in the random oracle, and thus outputs 1. So, with probability $(1 - \mathfrak{p}/(QQ')) \cdot 1/2 + \mathfrak{p}/(QQ') = 1/2 + \mathfrak{p}/(2QQ')$, $\mathcal{B}$ will output 1 when the oracle returns outputs of a pseudorandom function. In contrast, if the oracle returns outputs of a random function, $\hat{\boldsymbol{t}}$ is either used, in which case $(\hat{\mu}, \hat{\boldsymbol{t}})$ cannot be a bad ticket, or is hidden to $\mathcal{A}$. In the latter case, as $\hat{\boldsymbol{t}}$ is sampled uniformly in the range of the random function, $\mathcal{A}$ can guess $\hat{\boldsymbol{t}}$ with only a negligible probability. So, if the oracle returns outputs of a random function, $\mathcal{B}$ outputs 1 with a probability $1/2 + \mathit{negl}(\lambda)$. In summary, the advance of $\mathcal{B}$ is $\mathfrak{p}/(2QQ') - \mathit{negl}(\lambda)$, which is non-negligible.

• *Game 3.* This is identical to Game 2 except that when checking whether $\mathcal{A}$ succeeds, the challenger additionally attempts to extract the witness $(\boldsymbol{s}, \boldsymbol{y}, \sigma)$ from each proof Π in an authentication protocol transcript in the "bad series", and outputs 0 (indicating that $\mathcal{A}$ fails) if it fails to extract a valid witness for some proof Π in the "bad series", where we say a witness is valid if it satisfies the statement defiend in the authentication protocol in the construction. Indistinguishability between Game 2 and Game 3 comes from the simulation extractability property of the underlying ZKAoK systems directly, since each proof in the "bad series" is generated with an honest services provider, who will send a uniform challenge message $\mathfrak{m}$, which equals to a previously used challenge message with only a negligible probability. Thus, we have $| P_3 - P_2 | \leq \mathit{negl}(\lambda)$.

• *Game 4.* This is identical to Game 3 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \sigma)$, $\boldsymbol{y}$ has been queried to neither the $\mathcal{REG}$ oracle nor the $\mathcal{REG}_u$ oracle. Indistinguishability between Game 3 and Game 4 comes from the unforgeability of the underlying CL signature scheme directly. Thus, we have $| P_4 - P_3 | \leq \mathit{negl}(\lambda)$.

• *Game 5.* This is identical to Game 4 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \sigma)$, $\boldsymbol{y}$ has not been queried to the $\mathcal{REG}_u$ oracle. Game 3 and Game 4 are identical unless there ex-

ists an extracted witness $(\boldsymbol{s}, \boldsymbol{y}, \sigma)$ that y is the public key of an honest user. This can occur with only a negligible probability due to Lemma F.1. Thus, we have $\mid P_5 - P_4 \mid \leq negl(\lambda)$.

• *Game 6.* This is identical to Game 5 except that when checking whether $\mathcal{A}$ succeeds, the challenger outputs 0 if for two extracted witness $(\boldsymbol{s}_1, \boldsymbol{y}_1, \sigma_1)$ and $(\boldsymbol{s}_2, \boldsymbol{y}_2, \sigma_2)$, $\boldsymbol{y}_1 = \boldsymbol{y}_2$ but $\boldsymbol{s}_1 \neq \boldsymbol{s}_2$. Indistinguishability between Game 5 and Game 6 comes from the uniqueness of the underlying wPRF with efficient protocols directly. Thus, we have $\mid P_6 - P_5 \mid \leq negl(\lambda)$.

Note that in Game 6, if the adversary succeeds, it only use at most $\|\mathcal{U}_u\|$ different secret keys. Also, each of these $\|\mathcal{U}_u\|$ secret keys can only be used once unless a $\mathcal{REMOVE}$-$\mathcal{BL}$ oracle query is made to remove the ticket of that secret key from the blacklist of the target service provider. Also, each $\mathcal{REMOVE}$-$\mathcal{BL}$ oracle query can only enable the saved secret key to be used one more time since once it is used, it will be put into the blacklist again. So in Game 6, the adversary cannot success, i.e. $P_6 = 0$. That completes the proof of blacklistability.

Anonymity. Next, we prove the anonymity of the blacklistable anonymous credential system via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.

• *Game 0.* This is the real anonymity experiment defined in Appendix E.3.

• *Game 1.* This is identical to Game 0 except that when answering an authentication oracle on behalf of an honest user (i.e. when answering the oracle $\mathcal{AUTH}$ or the oracle $\mathcal{AUTH}_s$), the challenger generates a simulated proof Π. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $\mid P_1 - P_0 \mid \leq negl(\lambda)$. It is worth noting that in Game 1, the response $(\mu, \boldsymbol{t}, \Pi)$ in an authentication event involving an honest user can be generated with merely the knowledge of the public parameter, the blacklist of the counterpart service provider and the ticket $(\mu, \boldsymbol{t})$.

• *Game 2.* This is identical to Game 1 except that when answering an authentication oracle on behalf of an honest user (i.e. when answering the oracle $\mathcal{AUTH}$ or the oracle $\mathcal{AUTH}_s$), the challenger checks whether the user is blocked by the received blacklist via checking whether there exists a ticket generated by this user in the blacklist. Indistinguishability between Game 1 and Game 2 comes from the weak pseudorandomness of the underlying wPRF with efficient protocols, just as we have done in proving the blacklistability of the blacklistable anonymous credential system.

Note that in Game 2, the output of an authentication oracle involving an honest user is determined by the ticket generated this time (if have), the public information, and the history behaviors of the invloved honest user. Also, the output of a registeration oracle involving an honest user is determined by the public key of the user. Therefore, the view of $\mathcal{A}$ in Game 2 is in fact independent of the real value of these secret keys of honest users, and only depends on the functionalities of these secret keys.

• *Game 3.* This is identical to Game 2 except that public keys of honest users as well as each $\boldsymbol{t}$ in a ticket of an authentication event involving an honest user, is sampled uniformly from the range of F. Since during the whole procedure of Game 2 and Game 3, the challenger is only required to compute F on random inputs

and it will not be required to compute F on the same secret key/input pair twice, indistinguishability between Game 2 and Game 3 comes from the weak pseudorandomness of the underlying wPRF with efficient protocol directly. So, we have $| P_3 - P_2 | \leq negl(\lambda)$.
Note that in the query oracle of Game 3, for both $b = 0$ and $b = 1$, the check for whether the involved user is in the blacklist outputs the same result; also, the response (if have) is also independent of the bit b. Besides, the ticket $(\mu^*, \boldsymbol{t}^*)$ for the challenge oracle is not allowed to be used in any blacklist. Therefore, the view of $\mathcal{A}$ is independent of b, i.e. $P_3 = 1/2$. That completes the proof of the anonymity.

Non-Frameability. Next, we prove the non-frameability of the blacklistable anonymous credential system via defining the following games, where we use P_i to denote the probability that $\mathcal{A}$ succeeds in Game i.
• *Game 0.* This is the real non-frameability experiment defined in Appendix E.3.
• *Game 1.* This is identical to Game 0 except that when answering an authentication oracle on behalf of an honest user (i.e. when answering the oracle $\mathcal{AUTH}$ or the oracle $\mathcal{AUTH}_s$), the challenger generates a simulated proof Π. Indistinguishability between Game 0 and Game 1 comes from the zero-knowledge property of the underlying ZKAoK systems directly. So, we have $| P_1 - P_0 | \leq negl(\lambda)$. It is worth noting that in Game 1, the response $(\mu, \boldsymbol{t}, \Pi)$ in an authentication event involving an honest user can be generated with merely the knowledge of the public parameter, the blacklist of the counterpart service provider and the ticket $(\mu, \boldsymbol{t})$.
• *Game 2.* This is identical to Game 1 except that when answering an authentication oracle on behalf of an honest user (i.e. when answering the oracle $\mathcal{AUTH}$ or the oracle $\mathcal{AUTH}_s$), the challenger checks whether the user is blocked by the received blacklist via checking whether there exists a ticket generated by this user in the blacklist. Indistinguishability between Game 1 and Game 2 comes from the weak pseudorandomness of the underlying wPRF with efficient protocols, just as we have done in proving the blacklistability of the blacklistable anonymous credential system. Recall that we can in fact prove that the adversary is not able to put a bad ticket into a blacklist with a non-negligible probability in Game 2, where we say a ticket is bad if 1) $\mathsf{F}_s(\mathcal{H}(\mu^*)) = \boldsymbol{t}^*$ for a secret key $\boldsymbol{s}$ of an honest user and 2) $(\mu^*, \boldsymbol{t}^*)$ is not generated by that honest user. Therefore, the probability that $\mathcal{A}$ succeeds in Game 2 is negligible. That completes the proof of non-frameability.
□