# Concurrently Secure Identification Schemes and Ad Hoc Anonymous Identification Schemes Based on the Worst-Case Hardness of Lattice Problems

Akinori Kawachi[1], Keisuke Tanaka[1], and Keita Xagawa[1]

Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Japan.
E-mail:{kawachi,keisuke,xagawa5}@is.titech.ac.jp

**Abstract.** In this paper, we show that two variants of Stern's identification scheme [IEEE IT '96] are provably secure against concurrent attacks under the assumptions on the *worst-case* hardness of lattice problems. These assumptions are weaker than those for the existing schemes of Micciancio and Vadhan [CRYPTO '03] and of Lyubashevsky [PKC '08]. We also construct ad hoc anonymous identification schemes based on the lattice problems by modifying the variants.

**Keywords:** lattice-based cryptography, identification schemes, concurrent security, ad hoc anonymous identification schemes.

## 1 Introduction

Many researchers have so far developed cryptographic schemes based on combinatorial problems related to knapsacks, codes, and lattices, due to the intractability of the underlying problems, the efficiency of primitive operations, and the threat of quantum computers to number-theoretic schemes [22].

The cryptographic schemes based on combinatorial problems usually assume the *average-case* hardness of the underlying problem because they have to deal with randomly generated cryptographic instances such as keys, plaintexts, and ciphertexts. This implies security risk in such schemes since it is generally hard to show their average-case hardness. In fact, several attacks against such schemes, e.g., [21], were found in the practical settings. The cryptographic schemes only based on the average-case hardness are more likely to be at risk of these kinds of attacks.

It is therefore significant to guarantee the security under the worst-case hardness. Ajtai [2] showed that the average-case hardness of some lattice problem is equivalent to its worst-case hardness. His seminal result opened the way to cryptographic schemes based on the worst-case hardness of lattice problems. Several lattice-based schemes were proposed such as public-key encryption schemes, e.g., by Ajtai and Dwork [3], and hash functions [2,10,16].

Among varieties of lattice-based cryptographic schemes, there are very few results on the identification (ID) schemes based on the worst-case hardness of lattice problems. For example, Micciancio and Vadhan proposed ID schemes based on the worst-case hardness of lattice problems, such as the gap versions of the Shortest Vector Problem.

These schemes are obtained from their statistical zero-knowledge protocol with efficient provers [17]. Recently, Lyubashevsky also constructed lattice-based ID schemes secure against active attacks [12].

### 1.1 Our Contributions

In this paper, we propose two variants, which we call $S_{GL}$ and $S_{C/IL}$, of Stern's ID scheme [23]. The variants are secure against concurrent attacks[1] under the assumptions on the *worst-case* hardness of lattice problems, which are the gap version of the Shortest Vector Problem with an approximation factor $\tilde{O}(n)$ ($\mathrm{GapSVP}^2_{\tilde{O}(n)}$) and the Shortest Vector Problem for ideal lattices with an approximation factor $\tilde{O}(n)$ ($\Lambda(f)$-$\mathrm{SVP}^\infty_{\tilde{O}(n)}$), respectively, where $\tilde{O}(g(n)) = O(g(n)\operatorname{poly}\log g(n))$ for a function $g$ in $n$.

The gap version of SVP (GapSVP) has already been used in the known lattice-based cryptographic constructions [17,16,9]. SVP for ideal lattices ($\Lambda(f)$-SVP) was recently introduced by Lyubashevsky and Micciancio [13] to improve the efficiency of lattice-based hash functions along the line of Micciancio [14]. The efficiency of our ID scheme can be also improved by using SVP for ideal lattices.

Our variants basically follows the same framework as Stern's [23]. He presented a statistical zero-knowledge argument secure under the assumption of the average-case hardness of the Syndrome Decoding Problem (or a generalized version called the Modular Knapsack Problem) and a string commitment scheme. Changing the assumptions and adjusting the parameters, we successfully remove the assumption on a string commitment scheme since we can build it from the hash functions based on lattice problems.

Moreover, our variants yield efficient schemes for ad hoc anonymous identification (AID) based on the worst-case hardness of $\mathrm{GapSVP}^2_{\tilde{O}(n^2)}$ and $\Lambda(f)$-$\mathrm{SVP}^\infty_{\tilde{O}(n^2)}$, which are secure against concurrent chosen-group attacks. The AID scheme was originally proposed and formulated by Dodis, Kiayias, Nicolosi, and Shoup [7]. The protocol is done by two parties, a prover and verifier, but we implicitly suppose an ad hoc group. Given public keys of all members of the group to the verifier (and the prover), the goal is to convince the verifier that the prover belongs to the group, without being specified who the prover is of the group, if and only if the prover is an actual member of the group. Dodis et al. presented a general construction of AID schemes from any accumulator with one-way domain and showed that constant-size signer-ambiguous group and ring signatures can be obtained from AID schemes by using the Fiat-Shamir transformation.

We modify our variants of Stern's ID scheme into AID schemes based on a similar strategy to Wu, Chen, Wang, and Wang's [24], which gave an AID scheme based on the Weak Dependence Problem. However, there was neither formal definition of the assumption nor explicit security proof for their scheme in [24]. Thus, we formally define a concurrent version of the security notion, the security against impersonation under concurrent chosen-group attacks, and prove that our AID schemes have this security notion.

[1] In *active attacks*, an adversary could interact the prover prior to impersonation. In *concurrent attacks*, an adversary could interact many different prover "clones" concurrently prior to impersonation. Each clone has the same secret key, but has independent random coins and maintain its own state. After interacting many clones, the adversary would impersonate.

### 1.2 Overview of Our Variants

For simplicity, we only consider the ID scheme based on GapSVP in what follows. We first construct a string commitment scheme based on the lattice problem. Then we will describe the idea of the proof on concurrent security of the variant. Finally, we give a sketch of our construction method of an AID scheme.

Before giving overview, we review the fundamental problem, the Small Integer Solution Problem ($\mathrm{SIS}_{q,m,\beta}$), on which our variants are based; given a random $n$-by-$m$ matrix $\mathbf{A}$ whose elements are in $\mathbb{Z}_q$, the problem is finding an $m$-dimensional integral non-zero vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{0}$ and $\|\mathbf{x}\| \leq \beta$. Note that, informally, there exists a reduction from solving $\mathrm{GapSVP}^2_{\tilde{O}(\beta n^{1/2})}$ in the worst case to solving $\mathrm{SIS}_{q,m,\beta}$ on the average with non-negligible probability [16]. Also, we remark that the security of lattice-based hash functions inherits the hardness of SIS.

*String commitment scheme:* We construct a string commitment scheme from lattice-based hash functions. Meanwhile, general constructions of string commitment schemes from collision-resistant hash functions were shown by Damgård, Pedersen, and Pfizmann [5] and Halevi and Micali [11]. Stern also constructed a string commitment scheme from collision-resistant hash functions: Given a string $s$ and a random string $\rho$, a commitment is $h(\rho \circ (\rho \oplus s))$, where $\circ$ and $\oplus$ denote the concatenation and XOR operators, respectively, and $h$ is a hash function. We compose a string commitment scheme by more direct and simpler way than the general one and Stern's one: Given $s$ and $\rho$, a commitment is $h(\rho \circ s)$, where $h$ is a lattice-based hash function.

*Our ID scheme and its concurrent security:* In Stern's scheme and our variant, a prover has a binary vector $\mathbf{x}$ with a fixed Hamming weight as his/her secret key. We also feed to the prover and verifier a matrix $\mathbf{A}$ as a system parameter and a vector $\mathbf{y}$ as the public key corresponding to $\mathbf{x}$. The task of the prover is to convince the verifier that he/she knows a correct secret key $\mathbf{x}$ satisfying a relation $\mathbf{Ax} = \mathbf{y}$.

Stern proposed a statistical zero-knowledge argument for the above problem in [23]. First, the prover computes three commitments and sends them to the verifier. The verifier sends a random challenge to the prover. The prover reveals the commitments corresponding to the challenge. Stern constructed the knowledge extractor which computes a collision of a hash function in a string commitment scheme or a secret key corresponding to the target public key if a passive adversary responds correctly to any challenges after sending commitments,.

One of standard strategies to achieve concurrent security is to prove that a public key corresponds to multiple secret keys and the protocol is witness indistinguishable (WI) [8] and proof-of-knowledge: The reduction algorithm generates $sk$ and $pk$ and executes the adversary on $pk$ by simulating the prover with $sk$. Using the knowledge extractor of the protocol, the algorithm obtains another $sk'$ corresponding to $pk$ with certain probability since the protocol is WI. The algorithm then solves the underlying problem by using $pk$, $sk$, and $sk'$.

In our reduction, when the algorithm is given $\mathbf{A}$, it generates a secret key $\mathbf{x}$ and a public key $\mathbf{y} = \mathbf{Ax}$, and feeds $\mathbf{A}$ and $\mathbf{y}$ to the adversary. Note that the algorithm can simulate the prover with $\mathbf{A}$ and $\mathbf{x}$ that the adversary concurrently accesses. Using the

knowledge extractor for the adversary in Stern's proof, the algorithm obtains a collision of a string commitment scheme or a secret key $\mathbf{x}'$ such that $\mathbf{x} \neq \mathbf{x}'$ and $\mathbf{A}\mathbf{x}' = \mathbf{y}$, differently from the general strategy. In the former case, the algorithm outputs the collision of a hash function in the string commitment scheme which is based on $\mathrm{GapSVP}^2_{\tilde{O}(n)}$, and violates the assumption. In the latter case, the condition $\mathbf{x} \neq \mathbf{x}'$ will be satisfied by witness indistinguishability of Stern's protocol. Thus, the algorithm has the solution $\mathbf{z} = \mathbf{x} - \mathbf{x}'$ for SIS. Note that the $\ell_2$ norm of the solution is $\tilde{O}(n^{1/2})$. Thus, the assumption is the worst-case hardness of $\mathrm{GapSVP}^2_{\tilde{O}(n)}$.

*AID schemes:* Our construction for AID schemes also has a similar structure. Each of $l$ members in the ad hoc group has a vector $\mathbf{x}_i$ ($i = 1, \ldots, l$). Then, the common inputs of the scheme are a system parameter $\mathbf{A}$ and a set of public keys $\mathbf{y}_1, \ldots, \mathbf{y}_l$ of the members, which satisfy $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$ ($i = 1, \ldots, , l$). We can show that, by Stern's protocol, the prover can anonymously convince the verifier that the prover knows $\mathbf{x}_i$ corresponding to one of $\mathbf{y}_1, \ldots, \mathbf{y}_l$, since he/she knows a new vector $\mathbf{x}'$ such that $[\mathbf{A}\,\mathbf{y}_1 \ \ldots \ \mathbf{y}_l]\mathbf{x}' = \mathbf{0}$. This idea is due to Wu et al. [24]. We give security proof for the scheme, while they did not clearly give such a security proof for their scheme.

### 1.3 Comparison with Related Works

*ID schemes:* In [17], Micciancio and Vadhan proposed a statistical zero-knowledge and proof-of-knowledge protocol for $\mathrm{GapSVP}_\delta$ with some $\delta$. Combining it with lattice-based hash functions, we obtain an ID scheme which is secure against *passive attacks* where the underlying problem is $\mathrm{SIS}_{q,m,\tilde{O}(\delta n^{1/2})}$ and reduced from $\mathrm{GapSVP}^2_{\tilde{O}(\delta n)}$.

In the scheme, the prover and the verifier are given a matrix $\mathbf{A}$ as a common input, and the prover has a binary vector $\mathbf{x}$ as secret information. The task of the prover is to convince the verifier that he/she knows $\mathbf{x}$ satisfying the relation that $\mathbf{A}\mathbf{x} = \mathbf{0}$ and $\mathbf{x}$ is relatively short. It seems difficult to directly simulate the prover since a simulator has to prepare a dummy short vector $\mathbf{x}'$ satisfying $\mathbf{A}\mathbf{x}' = \mathbf{0}$, which is the task of SIS itself. Thus, we cannot straightforwardly prove the concurrent security for their ID scheme.

Actually, we can construct a concurrently secure ID scheme based on the worst-case hardness of lattice problems by Micciancio and Vadhan's ID scheme as noted in [17, Section 5]. Applying a technique of De Santis, Di Crescenzo, Persiano, and Yung [6] and a proof technique by Feige and Shamir [8], a modification of the ID scheme can be proven to have concurrent security[2]; we call it $\mathrm{MV}_{\mathrm{GL}}$ for short. We remark that the underlying problem is the same as the original scheme.

Recently, Lyubashevsky proposed new concurrently secure ID schemes based on lattice problems [12]; we call it $\mathrm{L}_{\mathrm{GL}}$ for short. In the $\mathrm{L}_{\mathrm{GL}}$ scheme, the underlying problem is $\mathrm{SIS}_{q,m,\tilde{O}(n)}$ and reduced from $\mathrm{GapSVP}^2_{\tilde{O}(n^{1.5})}$.

As mentioned in the previous section, the assumption of $\mathrm{S}_{\mathrm{GL}}$ is the worst-case hardness of $\mathrm{GapSVP}^2_{\tilde{O}(n)}$, which is weaker than those of $\mathrm{MV}_{\mathrm{GL}}$ and $\mathrm{L}_{\mathrm{GL}}$. See Table 1.

[2] Combining techniques by De Santis et al. [6] and Feige and Shamir [8], we derive a construction technique for ID schemes secure against active attacks. Moreover, by Bellare and Pracio's observation [4], we can construct concurrently secure ID schemes by the same technique.

| ID schemes ($\mathbf{A}_1, \mathbf{A}_2, \mathbf{A} \in \mathbb{Z}_q^{n\times m}$) | | | | | |
|---|---|---|---|---|---|
| | Sys. param. | Public key | Relation | $\gamma$ in $\mathrm{GapSVP}_\gamma^2$ | Comm. cost |
| $\mathrm{MV}_{\mathrm{GL}}$ [17] | – | $\mathbf{A}_1, \mathbf{A}_2$ | $\mathbf{A}_1\mathbf{x} = \mathbf{0}$ or $\mathbf{A}_2\mathbf{x} = \mathbf{0}$ | $\tilde{O}(\delta n)$ | $\tilde{O}(tk_\delta n)$ |
| $\mathrm{L}_{\mathrm{GL}}$ [12] | $(\mathbf{A})$ | $\mathbf{A}, \mathbf{y}$ | $\mathbf{A}\mathbf{x} = \mathbf{y}$ | $\tilde{O}(n^{1.5})$ | $\tilde{O}(tn)$ |
| $\mathrm{S}_{\mathrm{GL}}$ | $\mathbf{A}$ | $\mathbf{y}$ | $\mathbf{A}\mathbf{x} = \mathbf{y}$ and $w_H(\mathbf{x}) = w$ | $\tilde{O}(n)$ | $\tilde{O}(tn)$ |

| AID schemes ($\mathbf{A}_{i,0}, \mathbf{A}_{i,1}, \mathbf{A} \in \mathbb{Z}_q^{n\times m}$) | | | | | |
|---|---|---|---|---|---|
| Base | Sys. param. | Set of public keys | Relation | $\gamma$ in $\mathrm{GapSVP}_\gamma^2$ | Comm. cost |
| $\mathrm{MV}_{\mathrm{GL}}$ [17] | – | $\mathbf{A}_{1,0}, \mathbf{A}_{1,1}, \ldots, \mathbf{A}_{l,0}, \mathbf{A}_{l,1}$ | $\mathbf{A}_{i,0}\mathbf{x} = \mathbf{0}$ or $\mathbf{A}_{i,1}\mathbf{x} = \mathbf{0}$ | $\tilde{O}(\delta n)$ | $\tilde{O}(tlk_\delta n)$ |
| $\mathrm{L}_{\mathrm{GL}}$ [12] | $\mathbf{A}$ | $\mathbf{y}_1, \ldots, \mathbf{y}_l$ | $\mathbf{A}\mathbf{x} = \mathbf{y}_i$ | $\tilde{O}(n^{1.5})$ | $\tilde{O}(tln)$ |
| $\mathrm{S}_{\mathrm{GL}}$ | $\mathbf{A}$ | $\mathbf{y}_1, \ldots, \mathbf{y}_l$ | $\mathbf{A}\mathbf{x} = \mathbf{y}_i$ and $w_H(\mathbf{x}) = w$ | $\tilde{O}(\eta n^{1/2})$ | $\tilde{O}(t(l+n))$ |

**Table 1.** Comparisons among ID schemes and AID schemes. A secret key $sk$ is $\mathbf{x} \in \{0,1\}^m$. The factor $n$ denotes the security parameter. We denote the Hamming weight of $\mathbf{x}$ as $w_H(\mathbf{x})$. The protocol is run in $t$ times in parallel. The factor $\delta$ in $\mathrm{MV}_{\mathrm{GL}}$ is larger than $(m/\log m)^{1/2} = \tilde{O}(n^{1/2})$. The factor $k_\delta$ is $n^{O(1)}$ and $\omega(1)$, if $\delta$ is $\Omega(n^{1/2})$ and $n^{0.5+\Omega(1)}$, respectively. In the table below, $l$ denotes the number of the members in the group and the factor $\eta$ is $\max\{(w+1)^{3/2}, \sqrt{m}\}$.

*AID schemes:* By simple modifications of $\mathrm{MV}_{\mathrm{GL}}$ and $\mathrm{L}_{\mathrm{GL}}$, we could obtain AID schemes based on the worst-case hardness of lattice problems in a straightforward manner. We feed only $pk_0, \ldots, pk_l$ as the common inputs to the prover and the verifier. In this case, the prover convinces the verifier that he/she has a secret key corresponding to $pk_i$.

Unfortunately, each of these simple modifications requires a large overhead cost involving the size of the ad hoc group. Let $l$ be the number of the members of the group and $n$ the security parameter. The protocol is run in $t$ times in parallel to reduce the soundness error. The communication cost in the $\mathrm{MV}_{\mathrm{GL}}$-based scheme is $tl \cdot \tilde{O}(k_\delta n)$ (for $k_\delta$, see the caption in Table 1) and that in the $\mathrm{L}_{\mathrm{GL}}$-based scheme $tl \cdot \tilde{O}(n)$. The size of a set of the public keys is $l \cdot \tilde{O}(n^2)$ and $\tilde{O}(n^2) + l \cdot \tilde{O}(n)$ in the modified versions of $\mathrm{MV}_{\mathrm{GL}}$ and $\mathrm{L}_{\mathrm{GL}}$, respectively.

On AID schemes, our scheme and the $\mathrm{L}_{\mathrm{GL}}$-based scheme require many *vectors* proportional to the size of the group, while the $\mathrm{MV}_{\mathrm{GL}}$-based scheme requires many *matrices* proportional to the size of the group. (See Table 1.) Additionally, our scheme costs the communication $t \cdot \tilde{O}(n+l)$, while the $\mathrm{MV}_{\mathrm{GL}}$- and $\mathrm{L}_{\mathrm{GL}}$-based schemes cost $tl \cdot \tilde{O}(k_\delta)$ and $tl \cdot \tilde{O}(n)$, respectively. This shows the advantage of our scheme on the efficiency.

### 1.4 Organization

The rest of this paper is organized as follows. We introduce basic notations and notions, and review the cryptographic schemes we consider in Section 2. (The formal definitions are in Appendix A.) In Section 3, we review lattice-based hash functions and give a commitment scheme based on the lattice-based hash functions for our ID and AID schemes. In Section 4, we construct the ID scheme by combining the framework of

Stern's scheme with our string commitment scheme. We present the AID scheme in Section 5.

In this paper, due to lack of space, we only describe the schemes based on GapSVP since the construction from $\Lambda(f)$-SVP follows a similar strategy to that from GapSVP. We argue the constructions from $\Lambda(f)$-SVP in Appendix C.

## 2 Preliminaries

*Basic notions and notations:* We denote by $n$ the security parameter of cryptographic schemes throughout this paper, which corresponds to the rank of the underlying lattice problems. We say that a problem is hard in the worst case if there exists no probabilistic polynomial-time algorithm solves the problem in the worst case with a non-negligible probability. We sometimes use $\tilde{O}(g(n))$ for any function $g$ in $n$ as $O(g(n)\cdot\mathrm{polylog}(g(n)))$. We assume that all random variables are independent and uniform.

For any $p \geq 1$, the $\ell_p$ norm of a vector $\mathbf{x} = {}^t(x_1, \ldots, x_n) \in \mathbb{R}^n$, denoted by $\|\mathbf{x}\|_p$, is $(\sum_{i=1}^n x_i^p)^{1/p}$. For ease of notation, we define $\|\mathbf{x}\| := \|\mathbf{x}\|_2$. The $\ell_\infty$ norm is defined as $\|\mathbf{x}\|_\infty = \lim_{p\to\infty} \|\mathbf{x}\|_p = \max_i |x_i|$. Let $w_H(\mathbf{x})$ denote the Hamming weight of $\mathbf{x}$, i.e., the number of nonzero elements in $\mathbf{x}$. Let $\mathrm{B}(m, w)$ denote the set of binary vectors in $\{0,1\}^m$ whose Hamming weights are exactly equal to $w$, i.e., $\mathrm{B}(m, w) := \{\mathbf{x} \in \{0,1\}^m \mid w_H(\mathbf{x}) = w\}$. We denote the concatenation of two vectors or strings $\mathbf{v}_1$ and $\mathbf{v}_2$ as $\mathbf{v}_1 \circ \mathbf{v}_2$.

*Hash functions:* We briefly review the definition of collision-resistant hash functions family. Let $\mathcal{H}_n = \{h_k : M_n \to D_n \mid k \in K_n\}$ be a family of hash functions, where $M_n$, $D_n$, and $K_n$ denotes a space of messages, digests, and indices, respectively. Let $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$. Roughly speaking, if $\mathcal{H}$ is collision resistant, any polynomial-time adversary cannot, on input a random index $k$, output a collision of the hash function indexed by $k$. The formal definition is in Appendix A.1

*String commitment schemes:* We consider a string commitment scheme in the trusted setup model. The trusted setup model is often required to construct practically efficient cryptographic schemes such as non-interactive string commitment schemes. In this model, we assume that a trusted party $\mathcal{T}$ honestly sets up a system parameter for the sender and the receiver.

First $\mathcal{T}$ distributes the index $k$ of a commitment function to the sender and the receiver. Both parties then share a common function $\mathrm{Com}_k$ by a given $k$. The scheme executes two phase, called committing and revealing phases. In the committing phase, the sender commits his/her decision, say a string $s$, to a commitment string $c = \mathrm{Com}_k(s;\rho)$ with a random string $\rho$ and sends $c$ to the receiver. In the revealing phase, the sender gives the receiver the decision $s$ and the random string $\rho$. The receiver verifies the validity of $c$ by computing $\mathrm{Com}_k(s;\rho)$.

We require two security notions of the string commitment schemes, statistical-hiding and computational-binding properties. Intuitively, if $C$ is statistically hiding, any computationally unbounded adversarial receiver cannot distinguish two commitment strings generated from two distinct strings. Also, it is computationally hiding, any polynomial-time adversarial sender cannot change the committed string after sending the commitment. For the formal definition, see Appendix A.2

*Canonical identification schemes:* We adopt the definition of identification schemes given in [1]. Let $\mathcal{SI} = (\mathsf{SetUp}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$ be an identification scheme, where $\mathsf{SetUp}$ is the setup algorithm which on input $1^n$ outputs *param*, $\mathsf{KG}$ is the key-generation algorithm which on input *param* outputs $(pk, sk)$, $\mathsf{P}$ is the prover algorithm taking input $sk$, $\mathsf{V}$ is the verifier algorithm taking inputs *param* and $pk$. We say $\mathcal{SI}$ is a canonical identification scheme if it is a public-coin 3-move protocol.

We are interested in concurrent attacks, which are stronger than active and passive attacks. We employ the definition of concurrent security in [4].

In concurrent attacks, the adversary will play the role of a cheating verifier prior to impersonation and can interact many different prover clones concurrently. Each clone has the same secret key, but has independent random coins and maintains its own state. We say $\mathcal{SI}$ is secure against impersonation under concurrent attacks, if any polynomial-time adversary cannot, given a random public key of a legitimate prover, impersonate the legitimate prover. The formal definition is in Appendix A.3.

*Ad hoc anonymous identification schemes:* Dodis, Kiayias, Nicolosi, and Shoup introduced ad hoc anonymous identification schemes (AID schemes) [7], which are identification versions of ring signature schemes. Indeed, applying the Fiat-Shamir transform to their AID schemes, we can obtain ring signature schemes.

An AID scheme allows a user to anonymously prove his/her membership in a group if and only if the user is an actual member of the group, where the group is formed in an ad hoc fashion without help of the group manager. We then assume that every user registers his/her public key to the public key infrastructure.

We define the algorithms in AID schemes. An AID scheme is four tuple $\mathcal{AID} = (\mathsf{SetUp}, \mathsf{Reg}, \mathsf{P}, \mathsf{V})$, where $\mathsf{SetUp}$ is the setup algorithm which on input $1^n$ outputs *param*, $\mathsf{Reg}$ is the key generation and registration algorithm which on input *param* outputs $(pk, sk)$, $\mathsf{P}$ is the prover algorithm taking inputs *param*, a set of public keys $R = (pk_1, \ldots, pk_l)$, and one of the secret keys $sk_i$ such that $pk_i \in R$, and $\mathsf{V}$ is the verifier algorithm taking inputs *param* and $R$. We omit the group public-key and the group secret-key construction algorithms in the definition of [7] to simplify notations.

There are two goals for security of AID schemes: security against impersonation and anonymity.

Dodis et al. formally defined security against impersonation under passive attacks. They mentioned the definition of security against impersonation under concurrent attacks. However, they did not give the formal definition (See [7, Sec. 3.2]). Thus, we define the security notion with respect to concurrent attacks. In the setting of chosen-group attacks, the adversary could force the prover to prove the membership in an arbitrary group if the prover is indeed a member of the group. Additionally, concurrent attacks allow the cheating verifier to interact with the clones of any provers. Also, they allow the cheating prover to interact with the clones of provers, but prohibit it from interacting with the target provers. We say $\mathcal{AID}$ is secure against impersonation under concurrent chosen-group attacks, if any polynomial-time adversary cannot impersonate the legitimate prover in the above settings. The formal definition is in Appendix A.4.

The security notion, anonymity against full key exposure, captures the property that an adversary cannot distinguish two transcripts even if the adversary has the secret keys of all the members. We say $\mathcal{AID}$ is anonymous against full key exposure, any

polynomial-time adversary cannot distinguish two provers with a common set of public keys even if the adversary knows the secret keys of the two provers. Again, the formal definition is in Appendix A.4.

## 3 Main Tools

In this section, we review main tools, lattices, lattice problems, and lattice-based hash functions, and construct string commitment schemes.

*Lattices and lattice problems:* We first review fundamental notions of lattices, well-known lattice problems, and a related problem. An $n$-dimensional lattice in $\mathbb{R}^m$ is the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} \alpha_i \mathbf{b}_i \mid \alpha_i \in \mathbb{Z}\}$ of all integral combinations of $n$ linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$. The sequence of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is called a *basis* of the lattice $L$. We also denote $\mathbf{B}$ as the sequence of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$. For more details on lattices, see the textbook by Micciancio and Goldwasser [15].

We give the definitions of well-known lattice problems, the Shortest Vector Problem ($\mathrm{SVP}^p$) and its approximation version ($\mathrm{SVP}^p_\gamma$): The problem $\mathrm{SVP}^p$ is, given a basis $\mathbf{B}$ of a lattice $L$, finding the shortest non-zero vector $\mathbf{v}$ in $L$ in the $\ell_p$ norm. The problem $\mathrm{SVP}^p_\gamma$ is, given a basis $\mathbf{B}$ of a lattice $L$, finding a non-zero vector $\mathbf{v}$ in $L$ such that for any non-zero vector $\mathbf{x}$ in $L$, $\|\mathbf{v}\|_p \leq \gamma \|\mathbf{x}\|_p$.

We next give the definition of the gap version of $\mathrm{SVP}^p_\gamma$, which is the underlying problem of lattice-based hash functions [16].

**Definition 1** (**$\mathrm{GapSVP}^p_\gamma$**). *For a gap function $\gamma$, an instance of $\mathrm{GapSVP}^p_\gamma$ is a pair $(\mathbf{B}, d)$ where $\mathbf{B}$ is a basis of a lattice $L$ and $d$ is a rational number. In YES input there exists a vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$ such that $\|\mathbf{v}\|_p \leq d$. In NO input, for any vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$, $\|\mathbf{v}\|_p > \gamma d$.*

We also define the Small Integer Solution problem SIS (in the $\ell_p$ norm), which is often considered in the context of average-case/worst-case connections and a source of lattice-based hash functions as we see later.

**Definition 2** (**$\mathrm{SIS}^p_{q,m,\beta}$**). *For a fixed integer $q$ and real $\beta$, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the problem is finding a non-zero integer vector $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{Az} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\|_p \leq \beta$.*

*Lattice-based hash functions:* We review the lattice-based hash functions. Let $n$ be a security parameter (or a rank of an underlying lattice problem). For a prime $q = q(n) = n^{O(1)}$ and an integer $m = m(n) > n \log q(n)$, we define a family of hash functions, $\mathcal{H}(q, m) = \{ f_{\mathbf{A}} : \{0,1\}^m \to \mathbb{Z}_q^n \mid \mathbf{A} \in \mathbb{Z}_q^{n \times m}\}$, where $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{Ax} \bmod q$.

Originally, Ajtai [2] showed, for suitably chosen $q(n)$ and $m(n)$, the problem, which is, given $\mathcal{H}_{q,m}$, finding a short non-zero vector $\mathbf{v}$ in a lattice $\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} \equiv \mathbf{0} \pmod q\}$ such that $\|\mathbf{v}\| \leq n$, i.e., $\mathrm{SIS}^2_{q,m,n}$, is hard on the average under the assumption that $\mathrm{GapSVP}_\gamma$ is hard in the worst case within some polynomial approximation factor $\gamma$. It is known that $\mathcal{H}(q, m)$ is indeed collision resistant for suitably chosen $q$ and $m$ by Goldreich, Goldwasser, and Halevi [10]. They observed that finding a collision $(\mathbf{x}, \mathbf{x}')$ for $f_{\mathbf{A}} \in \mathcal{H}(q, m)$ implies finding a short non-zero vector $\mathbf{z} = \mathbf{x} - \mathbf{x}'$ such that $\|\mathbf{z}\| \leq \sqrt{m}$ and $\mathbf{Az} = \mathbf{0} \bmod q$, i.e., solving $\mathrm{SIS}^2_{q,m,\sqrt{m}}$. Recently, Micciancio and Regev showed that

$\mathcal{H}(q,m)$ is collision resistant under the assumption that $\mathrm{GapSVP}^2_{\tilde{O}(n)}$ is hard in the worst case [16].

**Theorem 1.** *For any polynomially bounded functions $\beta = \beta(n)$, $m = m(n)$, $q = q(n)$, with $q \geq 4\sqrt{m}n^{3/2}\beta$ and $\gamma = 14\pi\sqrt{n}\beta$, there exists a probabilistic polynomial-time reduction from solving $\mathrm{GapSVP}^2_\gamma$ in the worst case to solving $\mathrm{SIS}^2_{q,m,\beta}$ on the average with non-negligible probability.*

*Setup and key-generation algorithms:* We restrict the domain of hash functions to work the hash function for the key-generation algorithm in Stern's ID scheme. The restricted version is given as follows: $\mathcal{H}'(q,m,w) = \{h_{\mathbf{A}} : \mathrm{B}(m,w) \to \mathbb{Z}_q^n \mid \mathbf{A} \in \mathbb{Z}_q^{n\times m}\}$, where $h_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$. Observe that finding a collision $(\mathbf{x},\mathbf{x}')$ for $h_{\mathbf{A}} \in \mathcal{H}'(q,m,w)$ implies finding a short vector $\mathbf{z} = \mathbf{x} - \mathbf{x}'$ such that $\|\mathbf{z}\| \leq \min\{\sqrt{2w}, \sqrt{m}\}$ and $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$, i.e, solving the instance $(q, \mathbf{A}, \sqrt{m})$ of $\mathrm{SIS}^2_{q,m,\sqrt{m}}$.

The setup algorithm on input $1^n$ outputs a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. The key-generation algorithm on input $\mathbf{A}$ chooses a random vector $\mathbf{x} \in \mathrm{B}(m,w)$, computes $\mathbf{y} = \mathbf{A}\mathbf{x}$, and outputs $(pk, sk) = (\mathbf{y}, \mathbf{x})$.

*A string commitment scheme:* General constructions of statistically hiding and computationally binding string commitment schemes are known from a family of collision-resistant hash functions [5,11]. Their constructions used universal hash functions for the statistical-hiding property.

Here, we give a more direct and simpler construction from the lattice-based hash functions without the universal hash functions. The input of the commitment function is an $m$-bit vector $\mathbf{x}$ obtained by concatenating a random string $\rho = (\rho_1, \ldots, \rho_{m/2})$ and a message string $s = (s_1, \ldots, s_{m/2})$, i.e., $\mathbf{x} = \rho \circ s$. We then define the commitment function on inputs $s$ and $\rho$ as

$$\mathrm{Com}_{\mathbf{A}}(s;\rho) := \mathbf{A}\mathbf{x} \bmod q = \mathbf{A}^t(\rho_1, \ldots, \rho_{m/2}, s_1, \ldots, s_{m/2}) \bmod q.$$

**Lemma 1.** *For any polynomially bounded functions $m = m(n)$, $q = q(n)$, $\gamma = \gamma(n)$, with $q \geq 4mn^{3/2}$, $\gamma = 14\pi\sqrt{nm}$, and $m > 10n\log q$, $\mathrm{Com}_{\mathbf{A}}$ is a statistically hiding and computationally binding string commitment scheme in the trusted setup model if $\mathrm{GapSVP}^2_\gamma$ is hard in the worst case.*

The proof is in Appendix B.

Using the Merkle-Damgård technique, we obtain a string commitment scheme whose commitment function is $\mathrm{Com}_{\mathbf{A}} : \{0,1\}^* \times \{0,1\}^{m/2} \to \mathbb{Z}_q^n$ rather than $\mathrm{Com}_{\mathbf{A}} : \{0,1\}^{m/2} \times \{0,1\}^{m/2} \to \mathbb{Z}_q^n$. We use this commitment scheme in the rest of the paper. We often abuse the notation of $\mathrm{Com}_{\mathbf{A}}$. For example, we use the notation $\mathrm{Com}_{\mathbf{A}}(\mathbf{v}_1, \mathbf{v}_2; \rho)$, which is computed as follows: We first compute $s := \mathrm{string}(\mathbf{v}_1) \circ \mathrm{string}(\mathbf{v}_2)$, where a function "string" is outputs a binary string. Then, we compute $\mathrm{Com}_{\mathbf{A}}(s;\rho)$ as the commitment.

## 4 An Identification Scheme

Our variant $\mathrm{S}_{\mathrm{GL}}$ is obtained by plugging the string commitment scheme into Stern's ID scheme [23], which presents a zero-knowledge argument protocol based on the decoding problem on binary codewords called the Syndrome Decoding Problem. Since his

protocol deals with binary codewords, it works on the binary field $\mathbb{Z}_2$. Stern also proposed that an analogous scheme in $\mathbb{Z}_q$, where $q$ is extremely small number (typically 3, 5, or 7) [23, Section 6]. We adjust this parameter to connect his framework to our assumptions of the lattice problems.

The following is the ID scheme $S_{GL}$ based on GapSVP. Note that this protocol is parallelized in $t = \omega(\log n)$ times to achieve exponentially small soundness error.

SetUp: The setup algorithm, on input $1^n$, outputs a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ as *param*.

KG: The key-generation algorithm, on input $\mathbf{A}$, chooses a random vector $\mathbf{x} \in \mathrm{B}(m, w)$, computes $\mathbf{y} := \mathbf{A}\mathbf{x} \bmod q$. Outputs $(pk, sk) = (\mathbf{y}, \mathbf{x})$.

P, V: The common inputs are $\mathbf{A}$ and $\mathbf{y}$. The prover's auxiliary input is $\mathbf{x}$. They interact as follows:

**Step P1:** For every $i \in \{1, \ldots, t\}$, choose a random permutation $\pi_i$ over $\{1, \ldots, m\}$, a random vector $\mathbf{r}_i \in \mathbb{Z}_q^m$, and random strings $\rho_{i,1}, \rho_{i,2}$, and $\rho_{i,3}$. Compute $c_{i,1} := \mathrm{Com}_{\mathbf{A}}(\pi_i, \mathbf{A}\mathbf{r}_i; \rho_{i,1})$, $c_{i,2} := \mathrm{Com}_{\mathbf{A}}(\pi_i(\mathbf{r}_i); \rho_{i,2})$ and $c_{i,3} := \mathrm{Com}_{\mathbf{A}}(\pi_i(\mathbf{x} + \mathbf{r}_i); \rho_{i,3})$. Send $Cmt := ((c_{1,1}, c_{1,2}, c_{1,3}), \ldots, (c_{t,1}, c_{t,2}, c_{t,3}))$ to V.

**Step V1** V sends random challenges $Ch = (Ch_1, \ldots, Ch_t) \in \{1, 2, 3\}^t$ to P.

**Step P2** Parse $Ch$ as $(Ch_1, \ldots, Ch_t)$.

1. If $Ch_i = 1$, P reveals $c_{i,2}$ and $c_{i,3}$. Set $Rsp_i := (\pi_i(\mathbf{x}), \pi_i(\mathbf{r}_i), \rho_{i,2}, \rho_{i,3})$.
2. If $Ch_i = 2$, P reveals $c_{i,1}$ and $c_{i,3}$. Set $Rsp_i := (\pi_i, \mathbf{x} + \mathbf{r}_i, \rho_{i,1}, \rho_{i,3})$.
3. If $Ch_i = 3$, P reveals $c_{i,1}$ and $c_{i,2}$. Set $Rsp_i := (\pi_i, \mathbf{r}_i, \rho_{i,1}, \rho_{i,2})$.

Set $Rsp := (Rsp_1, \ldots, Rsp_t)$ and send $Rsp$ to V.

**Step V2** Parse $Rsp$ as $(Rsp_1, \ldots, Rsp_t)$.

1. If $Ch_i = 1$, parse $Rsp_i$ as $(\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \rho_{i,2}, \rho_{i,3})$. Check whether the weight of $\mathbf{x}$ and the commitments $c_{i,2}$ and $c_{i,3}$ are correct, that is, $\mathbf{z}_{i,1} \in \mathrm{B}(m, w)$, $c_{i,2} = \mathrm{Com}_{\mathbf{A}}(\mathbf{z}_{i,2}; \rho_{i,2})$, and $c_{i,3} = \mathrm{Com}_{\mathbf{A}}(\mathbf{z}_{i,1} + \mathbf{z}_{i,2}; \rho_{i,3})$. If they are correct, set $Dec_i := 1$, otherwise set $Dec_i := 0$.
2. If $Ch_i = 2$, parse $Rsp_i$ as $(\pi_i, \mathbf{z}_i, \rho_{i,1}, \rho_{i,3})$. Check whether the commitments $c_{i,1}$ and $c_{i,3}$ are correct, that is, $c_{i,1} = \mathrm{Com}_{\mathbf{A}}(\pi_i, \mathbf{A}\mathbf{z}_i - \mathbf{y}; \rho_{i,1})$ and $c_{i,3} = \mathrm{Com}_{\mathbf{A}}(\pi_i(\mathbf{z}_i); \rho_{i,3})$. If they are correct, set $Dec_i := 1$, otherwise set $Dec_i := 0$.
3. If $Ch_i = 3$, parse $Rsp_i$ as $(\pi_i, \mathbf{z}_i, \rho_{i,1}, \rho_{i,2})$. Check whether the commitments $c_{i,1}$ and $c_{i,2}$ are correct, that is, $c_{i,1} = \mathrm{Com}_{\mathbf{A}}(\pi_i, \mathbf{A}\mathbf{z}_i; \rho_{i,1})$ and $c_{i,2} = \mathrm{Com}_{\mathbf{A}}(\pi_i(\mathbf{z}_i); \rho_{i,2})$. If they are correct, set $Dec_i := 1$, otherwise set $Dec_i := 0$.

If $Dec_i = 1$ for all $i$, set $Dec := 1$, otherwise set $Dec := 0$. Output $Dec$.

We can show the theorem of the security on our ID protocol, which concerns impersonation under concurrent attacks. The proof is in Appendix B.

**Theorem 2.** *Assume that polynomially bounded functions $m = m(n)$ and $q = q(n)$ satisfy the condition that $q \geq 4mn^{3/2}$, $\gamma = 14\pi\sqrt{nm}$, $m \geq 10n \log q$, and $q^n / |\mathrm{B}(m, w)|$ is negligible in $n$. Then, the above ID scheme $S_{GL}$ is secure against impersonation under concurrent attacks if $\mathrm{GapSVP}_\gamma^2$ is hard in the worst case.*

*In particular, for any $m(n) = \Theta(n \log n)$, there exists $q(n) = O(n^{2.5} \log n)$, $\gamma(n) = O(n\sqrt{\log n})$, and $w(n) = \Theta(m(n))$ such that $q^n / |\mathrm{B}(m, w)|$ is negligible in $n$ and the ID scheme is secure against impersonation under concurrent attacks if $\mathrm{GapSVP}_\gamma^2$ is hard in the worst case.*

## 5 An Ad Hoc Anonymous Identification Scheme

The idea is as follows: Let $\mathbf{A}$ be a system parameter. Each user has a secret key $\mathbf{x}_i \in \mathrm{B}(m, w)$ and a public key $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$. In an AID scheme, a group is specified by a set of public keys $(\mathbf{y}_1, \ldots, \mathbf{y}_l)$ of the members. Let $\mathbf{e}_{i,l}$ denote an $l$-dimensional vector ${}^t(0, \ldots, 0, 1, 0, \ldots, 0)$ whose $i$-th element is 1. A user in the group, who has a secret key $\mathbf{x}_i$, convinces the verifier that he/she knows that $\mathbf{x}' := \mathbf{x}_i \circ -\mathbf{e}_{i,l}$ such that $[\mathbf{A}\,\mathbf{y}_1 \ \ldots \ \mathbf{y}_l]\mathbf{x}' = \mathbf{0}$, the number of 1 in $\mathbf{x}'$ is $w$, and the number of $-1$ in $\mathbf{x}'$ is 1.

We here construct an AID scheme based on GapSVP. We define $\mathrm{B}'(k, w)$ as $\{\mathbf{x} \in \{-1, 0, +1\}^k \mid w_{+1}(\mathbf{x}) = w \text{ and } w_{-1}(\mathbf{x}) = 1\}$, where $w_{+1}(\mathbf{x})$ denotes the number of $+1$ in $\mathbf{x}$ and $w_{-1}(\mathbf{x})$ denotes the number of $-1$ in $\mathbf{x}$.

Similarly to the ID scheme in Section 4, the protocol is repeated $t = \omega(\log n)$ times in parallel to achieve exponentially small soundness error. Due to lack of space, we only give the changes in the protocol.

SetUp: Same as SetUp in the protocol in Section 4.
Reg: Same as KG in the protocol in Section 4.
P, V: The common inputs are $\mathbf{A}$ and $(\mathbf{y}_1, \ldots, \mathbf{y}_l)$. The prover's auxiliary input is $sk_i = \mathbf{x}_i$. Let $\mathbf{A}' := [\mathbf{A}\,\mathbf{y}_1 \ \ldots \ \mathbf{y}_l] \in \mathbb{Z}_q^{n \times (m+l)}$ and $\mathbf{x}' := \mathbf{x}_i \circ -\mathbf{e}_{i,l}$. The prover would convince to verifier that he/she has $\mathbf{x}'$ such that $\mathbf{A}'\mathbf{x} = \mathbf{0}$ and $\mathbf{x}' \in \mathrm{B}'(m + l, w)$. Thus, the changes in the protocol is the followings: $m$, $\mathbf{A}$, $\mathbf{y}$, $\mathbf{x}$, and $\mathrm{B}(m, w)$ are replaced by $m + l$, $\mathbf{A}'$, $\mathbf{0}$, $\mathbf{x}'$, and $\mathrm{B}'(m + l, w)$, respectively.

The security of the above protocol is stated as follows. The proof is in Appendix B.3.

**Theorem 3.** *Assume that polynomially bounded functions $m = m(n)$, $q = q(n)$, and $w = w(n)$, satisfy the conditions that $m \geq 10n \log q$ and $q^n / |\mathrm{B}(m, w)|$ is negligible in $n$. Let $\beta := \max\{(w+1)^{3/2}, \sqrt{m}\}$. Assume that there exists an impersonator $\mathcal{I}$ that succeeds impersonation under concurrent chosen-group attacks with non-negligible probability. Then there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ that solves $\mathrm{SIS}^2_{q,m,\beta}$.*

Combining Theorem 3 with Theorem 1, we obtain the following theorem.

**Theorem 4.** *For any $m(n) = \Theta(n \log n)$, there exists $q(n) = O(n^{3.5} \log n)$, $\gamma(n) = O(n^2 \log^{3/2} n)$, and $w(n) = \Theta(m(n))$ such that $q^n / |\mathrm{B}(m, w)|$ is negligible in $n$ and the above scheme is secure against impersonation under concurrent chosen-group attacks if $\mathrm{GapSVP}^2_\gamma$ is hard in the worst case.*

The statistical anonymity of the above scheme follows from the witness-indistinguishability of Stern's protocol.

## References


1. Abdalla, M., An, J. H., Bellare, M., and Namprempre, C. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT 2002*, pp. 418–433.
2. Ajtai, M. Generating hard instances of lattice problems (extended abstract). In *STOC '96*, pp. 99–108. See also ECCC TR96-007.

3. Ajtai, M., and Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC '97*, pp. 284–293. See also ECCC TR96-065.
4. Bellare, M., and Palacio, A. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO 2002*, pp. 162–177.
5. Damgård, I. B., Pedersen, T. P., and Pfizmann, B. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology 10*, 3 (1997), 163–194.
6. De Santis, A., Di Crescenzo, G., Persiano, G., and Yung, M. On monotone formula closure of SZK. In *FOCS '94*, pp. 454–465.
7. Dodis, Y., Kiayias, A., Nicolosi, A., and Shoup, V. Anonymous identification in *ad hoc* groups. In *EUROCRYPT 2004*, pp. 609–626.
8. Feige, U., and Shamir, A. Witness indistinguishable and witness hiding protocols. In *STOC '90*, pp. 416–426.
9. Gentry, C., Peikert, C., and Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions, 2008. To appear, *STOC 2008*. See also ECCC TR07-133.
10. Goldreich, O., Goldwasser, S., and Halevi, S. Collision-free hashing from lattice problems. *ECCC 3*, 42 (1996).
11. Halevi, S., and Micali, S. Practical and provably-secure commitment scheme from collision-free hashing. In *CRYPTO '96*, pp. 201–215.
12. Lyubashevsky, V. Lattice-based identification schemes secure under active attacks, 2008. To appear, *PKC 2008*.
13. Lyubashevsky, V., and Micciancio, D. Generalized compact knapsacks are collision resistant. In *ICALP 2006*, pp. 144–155.
14. Micciancio, D. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity 16* (2007), 365–411. See also ECCC TR04-095.
15. Micciancio, D., and Goldwasser, S. *Complexity of Lattice Problems: a cryptographic perspective*. Kluwer Academic Publishers, 2002.
16. Micciancio, D., and Regev, O. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing 37*, 1 (2007), 267–302.
17. Micciancio, D., and Vadhan, S. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *CRYPTO 2003*, pp. 282–298.
18. Ohta, K., and Okamoto, T. On concrete security treatment of signatures derived from identification. In *CRYPTO '98*, pp. 354–369.
19. Peikert, C., and Rosen, A. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC 2006*, pp. 145–166.
20. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*, pp. 84–93.
21. Shamir, A. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Transactions on Information Theory 30*, 5 (1984), 699–704.
22. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing 26*, 5 (1997), 1484–1509.
23. Stern, J. A new paradigm for public key identification. *IEEE Transactions on Information Theory 42*, 6 (November 1996), 749–765.
24. Wu, Q., Chen, X., Wang, C., and Wang, Y. Shared-key signature and its application to anonymous authentication in ad hoc group. In *ISC 2004*, pp. 330–341.

# A Formal Definitions

*Provers and verifiers:* An interactive algorithm $\mathsf{A}$ is a stateful algorithm that, on input an incoming message $M_{in}$ and state information $St$, outputs an outgoing message $M_{out}$ and updated state $St'$ (we will write it as $(M_{out}, St') \leftarrow \mathsf{A}(M_{in}, St)$). We say that $\mathsf{A}$ accepts if $St = 1$ and rejects if $St = 0$.

An interaction between a prover $\mathsf{P}$ and a verifier $\mathsf{V}$ ends when $\mathsf{V}$ either accepts or rejects. We will write

$$(Tr, Dec) \leftarrow \mathbf{Run}[\mathsf{P}(p_1, \ldots)^{OP_1,\ldots} \leftrightarrow \mathsf{V}(v_1, \ldots)^{OV_1,\ldots}]$$

to indicate that we let $\mathsf{P}$ interact with $\mathsf{V}$, having provided both $\mathsf{P}$ and $\mathsf{V}$ with fresh random coins, to get a transcript $Tr$ and a boolean decision $Dec$.

## A.1 Hash Functions

We briefly review the definition of a family of collision-resistant hash functions. Let $\mathcal{H}_n = \{h_k : M_n \to D_n \mid k \in K_n\}$ be a family of hash functions and $\mathcal{H} := \{\mathcal{H}_n\}_{n\in\mathbb{N}}$.

We define the following experiment $\mathbf{Exp}^{\mathrm{col}}_{\mathcal{H},\mathcal{A}}(n)$ between the challenger and the adversary $\mathcal{A}$ for the collision-resistant property of a hash function.

**Experiment $\mathbf{Exp}^{\mathrm{col}}_{\mathcal{H},\mathcal{A}}(n)$:**

1. The challenger chooses $k$ from $K_n$ uniformly at random. The adversary $\mathcal{A}$ is given the security parameter $1^n$ and the system parameter $k$.
2. The adversary outputs $(x, x')$.
3. If $x, x' \in M_n$, $x \neq x'$, and $h_k(x) = h_k(x')$ then the challenger returns 1, otherwise, 0.

**Definition 3.** *Let $\mathcal{H}_n = \{h_k : M_n \to D_n \mid k \in K_n\}$ be a family of hash functions. Let $\mathcal{H} := \{\mathcal{H}_n\}_{n\in\mathbb{N}}$. Let $\mathcal{A}$ be an adversary. Let the advantage of $\mathcal{A}$ be $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{col}}_{\mathcal{H},\mathcal{A}}(n) = 1\right]$. We say that $\mathcal{H}$ is collision resistant if, for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},\mathcal{A}}(n)$ is negligible in $n$.*

## A.2 String Commitment Schemes

We consider a string commitment scheme in the trusted setup model. The trusted setup model is often required to construct practically efficient cryptographic schemes such as non-interactive string commitment schemes. In this model, we assume that a trusted party $\mathcal{T}$ honestly sets up a system parameter for a sender and receiver. In our case, the party $\mathcal{T}$ distributes a description of a commitment function randomly chosen from a family of commitment functions as the system parameter.

Let $C_n = \{\mathrm{Com}_k : M_n \times R_n \to C_n \mid k \in K_n\}$ be a family of commitment functions and let $C := \{C_n\}_{n\in\mathbb{N}}$. First, $\mathcal{T}$ on input $1^n$ distributes the system parameter $k \in K_n$ to a sender and receiver. Both parties then share a common function by $k$. After sharing the function $\mathrm{Com}_k$, the scheme executes two phases, called committing and revealing phases. In the committing phase, the sender commits his/her decision, say a string $s \in M_n$, to a

commitment string $c := \mathrm{Com}_k(s;\rho)$ with a random string $\rho \in R_n$. He/She then sends the commitment string $c$ to the receiver. In the revealing phase, the receiver verifies the sender's decision $s$ in the committing phase. To do so, the sender gives the receiver the decision $s$ and the random string $\rho$. The receiver can then easily verify the validity of $c$ by computing $\mathrm{Com}_k(s;\rho)$.

To define the security notion, consider the experiment $\mathbf{Exp}^{\mathrm{bd}}_{C,\mathcal{A}}(n)$ between the challenger and the adversary $\mathcal{A}$.

**Experiment $\mathbf{Exp}^{\mathrm{bd}}_{C,\mathcal{A}}(n)$:**

1. The challenger chooses $k$ from $K_n$ uniformly at random. The adversary $\mathcal{A}$ is given the security parameter $1^n$ and the system parameter $k$.
2. The adversary outputs $((s,\rho),(s',\rho'))$.
3. If $s, s' \in M_n$, $s \neq s'$, and $\mathrm{Com}_k(s;\rho) = \mathrm{Com}_k(s';\rho')$ then the challenger returns 1, otherwise, 0.

Here, the security notion of the string commitment schemes we require can be formalized as follows:

**Definition 4.** *We say a string commitment scheme $C$ is statistically hiding and computationally binding if it has the following properties:*

**Statistical-hiding property:**
*For any two strings $s, s' \in M_n$, the statistical distance between $\mathrm{Com}_k(s;\rho)$ and $\mathrm{Com}_k(s';\rho')$ is negligible in $n$ for random strings $\rho$ and $\rho'$.*

**Computational-binding property:**
*Let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}^{\mathrm{bd}}_{C,\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{bd}}_{C,\mathcal{A}}(n) = 1\right]$. Then, $\mathbf{Adv}^{\mathrm{bd}}_{C,\mathcal{A}}(n)$ is negligible in $n$ for any probabilistic polynomial-time adversary $\mathcal{A}$.*

Intuitively, if $C$ is statistically hiding, any computationally unbounded adversarial receiver cannot distinguish two commitment strings generated from two distinct strings. Also, it is computationally hiding, any polynomial-time adversarial sender cannot change the committed string after sending the commitment.

### A.3 Canonical Identification Schemes

We adopt the definition of identification schemes given in [1]. Let $\mathcal{SI} = (\mathsf{SetUp}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$ be an identification scheme, where $\mathsf{SetUp}$ is the setup algorithm which on input $1^n$ outputs *param*, $\mathsf{KG}$ is the key-generation algorithm which on input *param* outputs $(pk, sk)$, $\mathsf{P}$ is the prover algorithm taking input $sk$, $\mathsf{V}$ is the verifier algorithm taking inputs *param* and $pk$. We say $\mathcal{SI}$ is a canonical identification scheme if it is a public-coin 3-move protocol.

*Security against impersonation under concurrent attacks:* We are interested in concurrent attacks, which are stronger than active and passive attacks. So, we review the definition of concurrent security in [4]. In concurrent attacks, the adversary would play the role of a cheating verifier prior to impersonation, but could interact many different

| Experiment: $\mathbf{Exp}^{\text{imp-ca}}_{\mathcal{SI},\mathcal{I}}$ | |
|---|---|
| Input: | $n$ |
| Run: | 1. $param \leftarrow \mathsf{SetUp}(1^n)$ |
| | 2. $(pk, sk) \leftarrow \mathsf{KG}(param)$ |
| | 3. $PS \leftarrow \emptyset$ |
| | 4. $St_{\mathsf{CP}} \leftarrow \mathsf{CV}(1^n, param, pk)^{\text{Prov}}$ |
| | 5. $(Tr, Dec) \leftarrow \mathbf{Run}[\mathsf{CP}(St_{\mathsf{CP}}) \leftrightarrow \mathsf{V}(param, pk)]$ |
| Output: | $Dec$ |

| Prover oracle: Prov | |
|---|---|
| Input: | $s, M_{in}$ |
| Run: | 1. If $s \notin PS$ then |
| | 1-1. $PS \leftarrow PS \cup \{s\}$ |
| | 1-2. Pick a random coin $\rho$ for $\mathsf{P}$ |
| | 1-3. $St_{\mathsf{P}}[s] \leftarrow (param, sk, \rho)$ |
| | 2. $(M_{out}, St_{\mathsf{P}}[s]) \leftarrow \mathsf{P}(M_{in}, St_{\mathsf{P}}[s])$ |
| Output: | $M_{out}$ |

**Table 2.** Experiment and Oracle for Definition 5.

prover clones concurrently. Each clone has the same secret key, but has independent random coins and maintains its own state.

We describe the formal definition as follows. Consider the experiment $\mathbf{Exp}^{\text{imp-ca}}_{\mathcal{SI},\mathcal{I}}(n)$ between the challenger and the impersonator $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$.

**Experiment $\mathbf{Exp}^{\text{imp-ca}}_{\mathcal{SI},\mathcal{I}}(n)$:** (See also Table 2.)

**Setup Phase:** The challenger obtains $param \leftarrow \mathsf{SetUp}(1^n)$. Next, it obtains $(pk, sk) \leftarrow \mathsf{KG}(param)$ and sets $PS \leftarrow \emptyset$, where $PS$ denotes the set of prover's sessions. The impersonator $\mathsf{CV}$ is given the security parameter $1^n$, the system parameter $param$, and the target public key $pk$.

**Learning Phase:** The impersonator $\mathsf{CV}$ can query to the prover oracle Prov.

- The oracle Prov receives inputs $s, M_{in}$. If $s \notin PS$ then it adds $s$ to $PS$, picks a random coin $\rho$, and sets a state of the prover $St_{\mathsf{P}}[s] \leftarrow (param, sk, \rho)$. Next, it obtains $(M_{out}, St_{\mathsf{P}}[s]) \leftarrow \mathsf{P}(M_{in}, St_{\mathsf{P}}[s])$. It returns $M_{out}$.

**Challenge Phase:** $\mathsf{CV}$ outputs $St_{\mathsf{CP}}$. The challenger gives $St_{\mathsf{CP}}$ to $\mathsf{CP}$. Finally, the challenger obtains $(Tr, Dec) \leftarrow \mathbf{Run}[\mathsf{CP}(St_{\mathsf{CP}}) \leftrightarrow \mathsf{V}(param, pk)]$ and returns $Dec$.

**Definition 5.** *Let $\mathcal{SI} = (\mathsf{SetUp}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$ be an ID scheme, $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$ an impersonator, and n a security parameter. We define the advantage of $\mathcal{I}$ as $\mathbf{Adv}^{\text{imp-ca}}_{\mathcal{SI},\mathcal{I}}(n) := \Pr\left[\mathbf{Exp}^{\text{imp-ca}}_{\mathcal{SI},\mathcal{I}}(n) = 1\right]$. We say that $\mathcal{SI}$ is secure against impersonation under concurrent attacks if $\mathbf{Adv}^{\text{imp-ca}}_{\mathcal{SI},\mathcal{I}}(\cdot)$ is negligible for every polynomial-time $\mathcal{I}$.*

### A.4 Ad Hoc Anonymous Identification Schemes

An ad hoc anonymous identification (AID) scheme is four tuple $\mathcal{AID} = (\mathsf{SetUp}, \mathsf{Reg}, \mathsf{P}, \mathsf{V})$, where $\mathsf{SetUp}$ is the setup algorithm which on input $1^n$ outputs $param$, $\mathsf{Reg}$ is the

key generation and registration algorithm which on input *param* outputs $(pk, sk)$, $\mathsf{P}$ is the prover algorithm taking inputs *param*, a set of public keys $R = (pk_1, \ldots, pk_l)$, and one of secret key $sk_i$ such that $pk_i \in R$, $\mathsf{V}$ is the verifier algorithm taking inputs *param* and $R$. We omit the group public-key and the group secret-key construction algorithms in the definition of [7] to simplify notations. There are two goals for security of AID schemes: security against impersonation and anonymity.

*Security against impersonation under concurrent chosen-group attacks:* In the setting of chosen-group attacks, the adversary could force the prover to prove the membership in an arbitrary group if the prover is indeed a member of the group. Additionally, concurrent attacks allow the cheating verifier to interact with the clones of any provers. Also, they allow the cheating prover to interact with the clones of provers, but prohibit it from interacting with the target provers.

We describe the formal definition of the security as follows. Consider the following experiment $\mathbf{Exp}^{\text{imp-cca}}_{\mathcal{AID},\mathcal{I}}(n)$ between the challenger and the impersonator $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$.

**Experiment $\mathbf{Exp}^{\text{imp-cca}}_{\mathcal{AID},\mathcal{I}}(n)$:** (See also Table 3.)

**Setup Phase:** The challenger obtains $param \leftarrow \mathsf{SetUp}(1^n)$ and initializes $HU, CU, AU, PS \leftarrow \emptyset$, where $HU$, $CU$, and $TU$ denote the sets of honest users, corrupted users, and target users, respectively, and $PS$ denotes the set of prover's session. The impersonator $\mathsf{CV}$ is given the security parameter $1^n$ and the system parameter *param*.

**Learning Phase:** The impersonator $\mathsf{CV}$ can query to the three oracles Init, Corr, and Prov.

- The oracle Init receives input $i$. If $i \in HU \cup CU \cup TU$ then returns $\perp$. Otherwise, it obtains $(pk_i, sk_i) \leftarrow \mathsf{Reg}(param; r_i)$, adds $i$ to $HU$, and provides $\mathcal{I}$ with $pk_i$.
- The oracle Corr receives input $i$. If $i \notin HU \backslash TU$ then returns $\perp$. Otherwise, it adds $i$ to $CU$, deletes $i$ in $HU$, and returns $r_i$ to $\mathcal{I}$.
- The oracle Prov receives inputs $R = (pk_{i_1}, \ldots, pk_{i_l})$, $i$, $s$, and $M_{in}$. If $pk_i \notin R$ or $i \notin HU \backslash TU$ then returns $\perp$. (Note that the public keys in $R$ need not to be registered.) If $(R, i, s) \notin PS$ then it adds $(R, i, s)$ to $PS$, picks a random coin $\rho$, and sets a state of the prover $St_{\mathsf{P}}[(R, i, s)] \leftarrow (param, R, sk_i, \rho)$. Next, it obtains $(M_{out}, St_{\mathsf{P}}[(R, i, s)]) \leftarrow \mathsf{P}(M_{in}, St_{\mathsf{P}}[(R, i, s)])$. Finally, it returns $M_{out}$.

**Challenge Phase:** $\mathsf{CV}$ outputs a set of public keys $R_t = (pk_{i_1}, \ldots, pk_{i_l})$ and $St_{\mathsf{CP}}$. If the indices of the keys $\{i_1, \ldots, i_l\} \not\subseteq HU$ then the challenger outputs 0 and halts. Otherwise, the challenger sets $TU \leftarrow \{i_1, \ldots, i_l\}$ and gives $St_{\mathsf{CP}}$ to $\mathsf{CP}$. $\mathsf{CP}$ can query to the oracles Init, Corr, and Prov as in the learning phase. Finally, the challenger obtains $(Tr, Dec) \leftarrow \mathbf{Run}[\mathsf{CP}(St_{\mathsf{CP}})^{\text{Init,Corr,Prov}} \leftrightarrow \mathsf{V}(param, R_t)]$ and outputs *Dec*.

**Definition 6.** *Let $\mathcal{AID} = (\mathsf{SetUp}, \mathsf{Reg}, \mathsf{P}, \mathsf{V})$ be an AID scheme and $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$ an impersonator. Let $n$ be a security parameter. The advantage of $\mathcal{I}$ in attacking $\mathcal{AID}$ is defined by*

$$\mathbf{Adv}^{\text{imp-cca}}_{\mathcal{AID},\mathcal{I}}(n) := \Pr\left[\mathbf{Exp}^{\text{imp-cca}}_{\mathcal{AID},\mathcal{I}}(n) = 1\right].$$

*We say that $\mathcal{AID}$ is secure against impersonation under concurrent chosen-group attacks if* $\mathbf{Adv}^{\mathrm{imp-cca}}_{\mathcal{AID},\mathcal{I}}(\cdot)$ *is negligible for every polynomial-time $\mathcal{I}$.*

We note that our definition is the concurrent version of the soundness definition in [7].

| Experiment: $\mathbf{Exp}^{\mathrm{imp-cca}}_{\mathcal{AID},\mathcal{I}}$ | |
|---|---|
| Input: | $n$ |
| Run: | 1. $param \leftarrow \mathsf{SetUp}(1^n)$ |
| | 2. $HU, CU, TU, PS \leftarrow \emptyset$ |
| | 3. $(R_t, St_{\mathsf{CP}}) \leftarrow \mathsf{CV}(1^n, param)^{\textsc{Init},\textsc{Corr},\textsc{Prov}}$ |
| | 4. If $R_t \not\subseteq HU$ then return 0; |
| | 5. $TU \leftarrow R_t$ |
| | 5. $(Tr, Dec) \leftarrow \mathbf{Run}[\mathsf{CP}(St_{\mathsf{CP}})^{\textsc{Init},\textsc{Corr},\textsc{Prov}} \leftrightarrow \mathsf{V}(param, R_t)]$ |
| Output: | $Dec$ |

| User initiation oracle: Init | | User corruption oracle: Corr | |
|---|---|---|---|
| Input: | $i$ | Input: | $i$ |
| Run: | 1. If $i \in CU \cup HU \cup TU$ then return $\perp$ | Run: | 1. If $i \notin HU \setminus TU$ then return $\perp$ |
| | 2. $(pk_i, sk_i) \leftarrow \mathsf{Reg}(param; r_i)$ | | 2. $CU \leftarrow CU \cup \{i\}$ |
| | 3. $HU \leftarrow HU \cup \{i\}$ | | 3. $HU \leftarrow HU \setminus \{i\}$ |
| Output: | $pk_i$ | Output: | $r_i$ |

| Prover oracle: Prov | |
|---|---|
| Input: | $i, R, s, M_{in}$ |
| Run: | 1. If $pk_i \notin R$ then return $\perp$ |
| | 2. If $i \notin HU \setminus TU$ then return $\perp$ |
| | 3. If $(i, R, s) \notin PS$ then |
| | 3-1. $PS \leftarrow PS \cup \{(i, R, s)\}$ |
| | 3-2. Pick a random coin $\rho$ for $\mathsf{P}$ |
| | 3-3. $St_{\mathsf{P}}[i, R, s] \leftarrow (sk_i, R, \rho)$ |
| | 4. $(M_{out}, St_{\mathsf{P}}[i, R, s]) \leftarrow \mathsf{P}(M_{in}, St_{\mathsf{P}}[i, R, s])$ |
| Output: | $M_{out}$ |

**Table 3.** Experiment and oracles for Definition 6.

*Anonymity against full key exposure:* Anonymity against full key exposure for an AID scheme $\mathcal{AID}$ is defined by using the following experiment $\mathbf{Exp}^{\mathrm{anon-fke}}_{\mathcal{AID},\mathcal{A}}(n)$ between a challenger and adversary $\mathcal{A}$:

**Experiment $\mathbf{Exp}^{\mathrm{anon-fke}}_{\mathcal{AID},\mathcal{A}}(n)$:** (See also Table 4.)

**Setup Phase:** The challenger runs the algorithm $\mathsf{SetUp}$ with input $1^n$ and obtains $param$. The adversary $\mathcal{A}$ is given the system parameter $param$.

**Challenge Phase:** $\mathcal{A}$ requests a challenge by sending to the challenger the values $((pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1}), R)$. Here the set of public keys $R$ contains $pk_{i_0}$ and $pk_{i_1}$, and $(pk_{i_0}, sk_{i_0})$ and $(pk_{i_1}, sk_{i_1})$ are valid key pairs. The challenger chooses a random bit $b \in \{0, 1\}$ and runs the protocol as a prover who has $sk_{i_b}$. $\mathbf{Run}[\mathsf{P}(param, R, sk_{i_b}) \leftrightarrow \mathcal{A}]$.

**Output Phase:** $\mathcal{A}$ finally outputs its guess $b^*$ for $b$. If $b = b^*$ the challenger returns 1, otherwise returns 0.

**Definition 7.** *Let $\mathcal{AID} = (\mathsf{SetUp}, \mathsf{Reg}, \mathsf{P}, \mathsf{V})$ be an AID scheme, $\mathcal{A}$ an adversary, and $n$ a security parameter. The advantage of $\mathcal{A}$ in attacking $\mathcal{AID}$ is defined by*

$$\mathbf{Adv}^{\text{anon-fke}}_{\mathcal{AID},\mathcal{A}}(n) := \left|\Pr\left[\mathbf{Exp}^{\text{anon-fke}}_{\mathcal{AID},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

*We say that $\mathcal{AID}$ has anonymity with full key exposure if $\mathbf{Adv}^{\text{anon-fke}}_{\mathcal{AID},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time $\mathcal{A}$.*

| Experiment: $\mathbf{Exp}^{\text{anon-fke}}_{\mathcal{AID},\mathcal{A}}$ | |
|---|---|
| Input: | $n$ |
| Run: | 1. $param \leftarrow \mathsf{SetUp}(1^n)$ |
| | 2. $((pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1}), R, St_{\mathcal{A}}) \leftarrow \mathcal{A}(param)$. |
| | 3. $b \leftarrow_R \{0,1\}$. |
| | 4. $b^* \leftarrow \mathbf{Run}[\mathsf{P}(param, R, sk_{i_b}) \leftrightarrow \mathcal{A}(St_{\mathcal{A}})]$ |
| | 5. If $b = b^*$ then $Dec \leftarrow 1$. Otherwise $Dec \leftarrow 0$. |
| Output: | $Dec$ |

**Table 4.** Experiment and oracles for Definition 7.

# B Proofs

## B.1 Proof of Lemma 1

Before the proof, we review a definition of statistical distances: Given two probability density functions $\phi_1$ and $\phi_2$ on a finite set $S$, we define the statistical distance between them as $\Delta(\phi_1, \phi_2) := \frac{1}{2}\sum_{x\in S}|\phi_1(x) - \phi_2(x)|$. We also use the same notation for two arbitrary functions. Note that the acceptance probability of any algorithm on inputs from $X$ differs from its acceptance probability on inputs from $Y$ by at most $\Delta(X, Y)$.

*Proof.* The computational-binding property immediately follows from the collision-resistant property. We now show the statistical-hiding property.

Let $\mathbf{A} = [\mathbf{a}_1 \cdots \mathbf{a}_m]$. We then have $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \sum_{i=1}^{m/2}\rho_i\mathbf{a}_i + \sum_{i=1}^{m/2} s_i\mathbf{a}_{i+m/2}$. The following claim in [20] proves a random subset sum of $\mathbf{a}_i$ is statistically close to the uniform distribution.

*Claim.* Let $G$ be a finite Abelian group and let $l \geq c\log|G|$. If $c \geq 5$, $\Pr\left[\Delta\left(\left((g_1,\ldots,g_l), \sum_{i=1}^{l} r_i g_i\right), ((g_1,\ldots,g_l), u)\right) > \frac{2}{|G|}\right] \leq \frac{1}{|G|}$ for random variables $g_1,\ldots,g_l \in G, r_1,\ldots,r_l \in \{0,1\}$ and $u \in G$.

In our proof, we consider $\mathbb{Z}_q^n$ as the finite group $G$. Hence, we obtain that $\Delta\left(((\mathbf{a}_1,\ldots,\mathbf{a}_{m/2}),\sum_{i=1}^{m/2}\rho_i\mathbf{a}_i),((\mathbf{a}_1,\ldots,\mathbf{a}_{m/2}),\mathbf{u})\right)$ is then negligible with probability exponentially close to 1, where $\mathbf{u} \in \mathbb{Z}_q^n$ is a uniform random variable. Thus, $\Delta\left((\mathbf{A},\mathrm{Com}_{\mathbf{A}}(0^{m/2};\rho)),(\mathbf{A},\mathbf{u})\right)$ is negligible. Since $\Delta\left((\mathbf{A},\mathbf{u}+\sum_{i=1}^{m/2}s_i\mathbf{a}_{i+m/2}),(\mathbf{A},\sum_{i=1}^{m/2}\rho_i\mathbf{a}_i+\sum_{i=1}^{m/2}s_i\mathbf{a}_{i+m/2})\right)$ is also negligible, $\Delta((\mathbf{A},\mathrm{Com}_{\mathbf{A}}(s;\rho)),(\mathbf{A},\mathbf{u}))$ is negligible for any message $s$. By the triangle inequality, we have

$$\Delta((\mathbf{A},\mathrm{Com}_{\mathbf{A}}(s_1;\rho_1)),(\mathbf{A},\mathrm{Com}_{\mathbf{A}}(s_2;\rho_2))) \leq \Delta((\mathbf{A},\mathrm{Com}_{\mathbf{A}}(s_1;\rho_1)),(\mathbf{A},\mathbf{u}))+\Delta((\mathbf{A},\mathrm{Com}_{\mathbf{A}}(s_2;\rho_2)),(\mathbf{A},\mathbf{u}))$$

for any messages $s_1$ and $s_2$ and uniform random strings $\rho_1$ and $\rho_2$. It follows that $\Delta((\mathbf{A},\mathrm{Com}_{\mathbf{A}}(s_1;\rho_1)),(\mathbf{A},\mathrm{Com}_{\mathbf{A}}(s_2;\rho_2)))$ is negligible in $n$, which completes the proof. □

### B.2 Proof of Theorem 2

Before the proof of security, we need to mention the following trivial lemma.

**Lemma 2.** *For any fixed* $\mathbf{A}$*, let* $Y := \{\mathbf{y} \in \mathbb{Z}_q^n \mid |\{\mathbf{x} \in \mathrm{B}(m,w) \mid \mathbf{Ax} = \mathbf{y}\}| = 1\}$*, i.e., a set of vectors* $\mathbf{y}$ *such that the preimage* $\mathbf{x}$ *of* $\mathbf{y}$ *is uniquely determined for* $\mathbf{A}$*. If* $q^n/|\mathrm{B}(m,w)|$ *is negligible in n, then the probability that, if we obtain* $(\mathbf{y},\mathbf{x}) \leftarrow \mathsf{KG}(\mathbf{A})$*, then* $\mathbf{y} \in Y$ *is negligible in n.*

Note that, combining the witness-indistinguishability property of Stern's scheme with Lemma 2, we indeed show that our scheme has the witness-hiding property. We now prove Theorem 2.

*Proof.* We show that if there exists an impersonator $\mathcal{I}$ which succeeds impersonation under concurrent attacks with non-negligible probability $\epsilon$, there exists $\mathcal{A}$ that solves $\mathrm{SIS}^2_{q,m,\sqrt{m}}$ on the average. Then there exists any instance of $\mathrm{GapSVP}^2_\gamma$ by Theorem 1.

We first overview the strategy of $\mathcal{A}$. The algorithm $\mathcal{A}$ can control the impersonator $\mathcal{I}$ by feeding a random tape and a challenge. Given $\mathbf{A}$, $\mathcal{A}$ chooses a random secret key $\mathbf{x} \in \mathrm{B}(m,w)$ and computes $\mathbf{y} := \mathbf{Ax}$. We note that $\mathcal{A}$ can simulate the oracle Prov, since $\mathcal{A}$ has the secret key $\mathbf{x}$. $\mathcal{A}$ executes $\mathcal{I}$ on input $(\mathbf{A},\mathbf{y})$ three times with random challenges and a fixed random tape. Then, $\mathcal{A}$ obtains three transcripts $(Cmt^{(i)},Ch^{(i)},Rsp^{(i)},Dec^{(i)})$ for $i = 1,2,3$ as the results of the interactions between $\mathcal{I}$ and $\mathcal{A}$. Note that $Cmt^{(1)} = Cmt^{(2)} = Cmt^{(3)}$ since $\mathcal{A}$ fixes the random tape to work $\mathcal{I}$. By the assumption, $\mathcal{A}$ obtains good transcripts with non-negligible probability such that each $Dec^{(i)}$ is 1. Then, $\mathcal{A}$ can find the secret key $\mathbf{x}'$ corresponding to $\mathbf{y}$ or find $(s,\rho) \neq (s',\rho')$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathrm{Com}_{\mathbf{A}}(s';\rho')$ by using the fact that $Cmt^{(1)} = Cmt^{(2)} = Cmt^{(3)}$. In the former case, we will show that $\mathbf{x}' \neq \mathbf{x}$ with probability at least $1/2$. $\mathcal{A}$ outputs $\mathbf{z} = \mathbf{x}' - \mathbf{x}$. Since $\mathbf{z} \in \{-1,0,+1\}^m$, the norm $\|\mathbf{z}\| \leq \sqrt{m}$. In the latter case, $\mathcal{A}$ computes $\mathbf{z} \neq \mathbf{z}' \in \{0,1\}^m$ from $(s,\rho)$ and $(s',\rho')$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathbf{Az}$ and $\mathrm{Com}_{\mathbf{A}}(s';\rho') = \mathbf{Az}'$. Thus, $\mathcal{A}$ outputs $\mathbf{z}'' = \mathbf{z}' - \mathbf{z}$, where $\|\mathbf{z}''\| \leq \sqrt{m}$.

We now give the details. First, $\mathcal{A}$ executes the following procedure.

1. Choose a random tape $r$ of $\mathcal{I}$.

2. Choose challenges $Ch^{(1)}, Ch^{(2)}, Ch^{(3)}$ from $\{1,2,3\}^t$ uniformly at random.
3. For each $i = 1,2,3$, execute the experiment with random challenges $Ch^{(i)}$ and a fixed random tape $r$. $\mathcal{A}$ obtains three transcripts $(Cmt^{(i)}, Ch^{(i)}, Rsp^{(i)}, Dec^{(i)})$.

We have that the probability that all $Dec^{(i)}$ are 1 is at least $(\epsilon/2)^3$ by the Heavy Row Lemma [18]. Parse $Ch^{(i)}$ as $(Ch_1^{(i)}, \ldots, Ch_t^{(i)}) \in \{1,2,3\}^t$. We have $\Pr[\exists j \in \{1,\ldots,t\} : Ch_j^{(1)} \neq Ch_j^{(2)}, Ch_j^{(2)} \neq Ch_j^{(3)}, Ch_j^{(3)} \neq Ch_j^{(1)}] = 1 - (7/9)^t$ by a simple calculation. $\mathcal{A}$ therefore obtains good three transcripts with non-negligible probability $(\epsilon/2)^3 - (7/9)^t$, since $t = \omega(\log n)$.

We next show how $\mathcal{A}$ obtains a secret key or finding a collision of the hash functions in the string commitment scheme by using three good transcripts. Assume that $\mathcal{A}$ has three transcripts $(Cmt^{(i)}, Ch^{(i)}, Rsp^{(i)}, Dec^{(i)})$ for $i = 1,2,3$ such that $Cmt^{(1)} = Cmt^{(2)} = Cmt^{(3)}$, $Dec^{(i)} = 1$ for all $i$, and $\{Ch_j^{(1)}, Ch_j^{(2)}, Ch_j^{(3)}\} = \{1,2,3\}$ for some $j \in \{1,\ldots,t\}$. Without loss of generality, we assume that $Ch_j^{(i)} = i$. We parse $Rsp_j^{(i)}$ as in Step V2. From the above argument, we have four equations as follows (We omit $j$ for simplification):

$$
\begin{aligned}
c_1 &= \mathrm{Com}_{\mathbf{A}}(\pi^{(2)}, \mathbf{A}\mathbf{z}^{(2)} - \mathbf{y}; \rho_1^{(2)}) = \mathrm{Com}_{\mathbf{A}}(\pi^{(3)}, \mathbf{A}\mathbf{z}^{(3)}; \rho_1^{(3)}),\\
c_2 &= \mathrm{Com}_{\mathbf{A}}(\mathbf{z}_2^{(1)}; \rho_2^{(1)}) \qquad\qquad = \mathrm{Com}_{\mathbf{A}}(\pi^{(3)}(\mathbf{z}^{(3)}); \rho_2^{(3)}),\\
c_3 &= \mathrm{Com}_{\mathbf{A}}(\mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)}; \rho_3^{(1)}) \qquad = \mathrm{Com}_{\mathbf{A}}(\pi^{(2)}(\mathbf{z}^{(2)}); \rho_3^{(2)}),\\
\mathbf{z}_1^{(1)} &\in \mathrm{B}(m,w).
\end{aligned}
$$

If there exists a distinct pair of arguments of $\mathrm{Com}_{\mathbf{A}}$, $\mathcal{A}$ obtains a collision for $\mathbf{A}$ and solves $\mathrm{SIS}_{q,m,\sqrt{m}}$ as in the overview.

Next, we suppose that there exist no distinct pairs of the arguments of $\mathrm{Com}_{\mathbf{A}}$. Let $\pi$ denote the inverse permutation of $\pi^{(2)}$. From the first equation, we have $\pi^{-1} = \pi^{(2)} = \pi^{(3)}$. Thus, we obtain $\mathbf{z}^{(2)} = \pi(\mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)})$ from the third equation. Combining it with the first equation, we have $\mathbf{A}\mathbf{z}^{(3)} = \mathbf{A}(\pi(\mathbf{z}_1^{(1)}) + \pi(\mathbf{z}_2^{(1)})) - \mathbf{y}$. We obtain $\mathbf{y} = \mathbf{A}\pi(\mathbf{z}_1^{(1)})$ since $\pi^{-1}(\mathbf{z}_2^{(1)}) = \mathbf{z}^{(3)}$ in the second equation. We already have $\mathbf{z}_1^{(1)} \in \mathrm{B}(m,w)$. Then, $\mathcal{A}$ sets $\mathbf{x}' := \pi(\mathbf{z}_1^{(1)})$.

We now have to show that $\mathbf{x}' \neq \mathbf{x}$ with probability at least $1/2$. By Lemma 2, there must be another secret key $\mathbf{x}'$ corresponding to $\mathbf{y}$ with overwhelming probability. Note that the protocol is indeed a statistical witness-indistinguishable protocol and $\mathcal{I}$'s view is independent of $\mathcal{A}$'s choice of $\mathbf{x}$ with overwhelming probability, since Com is a statistically hiding string commitment scheme. Thus we have $\mathbf{x}' \neq \mathbf{x}$ with probability at least $1/2$. In this case $\mathcal{A}$ solves $\mathrm{SIS}_{q,m,\sqrt{m}}$ as in the overview. □

### B.3 Proof of Theorem 3

*Proof.* The algorithm $\mathcal{A}$, given input $\mathbf{A}$, feeds $\mathbf{A}$ to the impersonator $\mathcal{I}$. In the experiment, the impersonator $\mathcal{I}$ will call Init, Corr, and Prov. If $\mathcal{I}$ calls Init with input $i$, then $\mathcal{A}$ chooses $\mathbf{s}_i$ at random, computes $\mathbf{y}_i := \mathbf{A}\mathbf{s}_i$, and returns $\mathbf{y}_i$ to $\mathcal{I}$. Note that $\mathcal{A}$ can simulate the oracles Corr and Prov, since $\mathcal{A}$ has the secret key $\mathbf{s}_i$ corresponding to the public key $\mathbf{y}_i$.

At the end of the experiment, $\mathcal{I}$ will impersonate as a group which is specified by the set of the public keys $R = (\mathbf{y}_1, \ldots, \mathbf{y}_l)$. Rewinding $\mathcal{I}$ three times, $\mathcal{A}$ obtains

$(s,\rho) \neq (s',\rho')$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathrm{Com}_{\mathbf{A}}(s';\rho)'$ or a vector $\mathbf{x} = \mathbf{x}_1 \circ \mathbf{x}_2$ such that $[\mathbf{A}\,\mathbf{y}_1\ \ldots\ \mathbf{y}_l]\mathbf{x} = \mathbf{0}$, where $\mathbf{x}_1 \in \{-1,0,1\}^m$, $\mathbf{x}_2 \in \{-1,0,1\}^l$, and $\mathbf{x} \in \mathrm{B}'(m+l,w)$ as in the proof of Theorem 2.

In the former case, $\mathcal{A}$ computes $\mathbf{z} \neq \mathbf{z}' \in \{0,1\}^m$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathbf{A}\mathbf{z}$ and $\mathrm{Com}_{\mathbf{A}}(s';\rho') = \mathbf{A}\mathbf{z}'$. Hence, $\mathcal{A}$ can output $\mathbf{z}'' = \mathbf{z}' - \mathbf{z}$ such that $\|\mathbf{z}''\| \leq \sqrt{m}$.

In the latter case, we have $\mathbf{A}\mathbf{x}_1 + \sum_{i=1}^{l} x_{2,i}\mathbf{y}_i = \mathbf{0}$, that is, $\mathbf{A}\mathbf{x}_1 + \sum_{i=1}^{l} x_{2,i}\mathbf{A}\mathbf{s}_i = \mathbf{0}$. Hence, we obtain that $\mathbf{A}(\mathbf{x}_1 + \sum_{i=1}^{l} x_{2,i}\mathbf{s}_i) = \mathbf{0}$. Recall that the numbers of $+1$ in $\mathbf{x}$ is $w$ and that of $-1$ in $\mathbf{x}$ is 1. Hence, $\left\|\mathbf{x}_1 + \sum_{i=1}^{l} x_{2,i}\mathbf{s}_i\right\| \leq \|\mathbf{x}_1\| + \sum_{i=1}^{l} \left|x_{2,i}\right| \|\mathbf{s}_i\| \leq \sqrt{w+1} + (w+1)\sqrt{w} \leq (w+1)^{3/2}$. By the same argument as in the proof of Theorem 2, we have that $\mathbf{x}_1 + \sum_i x_{2,i}\mathbf{s}_i \neq \mathbf{0}$ with probability at least $1/2$. Thus, $\mathcal{A}$ outputs $\mathbf{z} := \mathbf{x}_1 + \sum_i x_{2,i}\mathbf{s}_i$ and solves $\mathrm{SIS}^2_{q,m,(w+1)^{3/2}}$ with non-negligible probability. □

# C Constructions from the Cyclic/Ideal Lattice Based Hash Functions

In this section, we construct the ID scheme and the AID scheme based on the cyclic/ideal lattice based hash functions. We basically follow the notations of [13].

## C.1 The Cyclic/Ideal Lattice Based Hash Functions

Several families of lattice-based hash functions are known to have small description sizes such as [14,19,13]. Originally, Micciancio [14] gave the compact version of the lattice-based hash functions and proved the one-wayness of the version. After that, Peikert and Rosen [19] and Lyubashevsky and Micciancio [13] proposed the modified versions of the version of Micciancio and showed their collision-resistance property, independently. We adopt the version of Lyubashevsky and Micciancio, since its generality of the descriptions.

Let $f \in \mathbb{Z}[x]$ be a monic and irreducible polynomial of degree $n$. Consider the quotient ring $\mathbb{Z}[x]/\langle f\rangle$. We use the standard set of representatives $\{(g \bmod f) : g \in \mathbb{Z}[x]\}$. In this section we identify a polynomial $\mathbf{a}(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \in \mathbb{Z}[x]/\langle f\rangle$ with an $n$-dimensional integer vector $\mathbf{a} = {}^t(a_0,\ldots,a_{n-1})$. We define a norm with respect to $f$ as follows: For $g \in \mathbb{Z}[x]$, $\|(g + \langle f\rangle)\|_f = \|g \bmod f\|_\infty$. We write $\|g\|_f$ instead of $\|g + \langle f\rangle\|_\infty$.

We note that any ideal $I \subseteq \mathbb{Z}[x]/\langle f\rangle$ defines the corresponding $n$-dimensional integer lattice $L(I) \subseteq \mathbb{Z}^n$. Notice that a class of the lattices representable in this way is contained in a general class of all integer lattices $L(\mathbf{B}) \subseteq \mathbb{Z}^n$. If a given lattice in $\mathrm{SVP}^p$ is restricted in a class $\Lambda$ of lattices, we denote by $\Lambda$-$\mathrm{SVP}^p$ the problem over such restricted lattices in $\Lambda$. We also denote by $\Lambda(f)$ the set of lattices that are isomorphic to ideals of $\mathbb{Z}[x]/\langle f\rangle$. See [13] for the details. We here deal with $\Lambda(f)$-$\mathrm{SVP}^\infty_\gamma$, i.e., SVP with approximation factor $\gamma$ in the $\ell_\infty$ norm whose input lattices are restricted in $\Lambda(f)$.

Lyubashevsky and Micciancio constructed a family of collision-resistant hash functions based on the worst-case hardness of $\Lambda(f)$-SVP for suitable $f$.

We review what $f$ is suitable for the construction of Lyubashevsky and Micciancio. The property of $f$ is defined as that the ring norm $\|g\|_f$ is not much bigger than $\|g\|_\infty$ for

any polynomial $g$. Formally, they captured this property as the *expansion factor* of $f$:

$$\mathrm{EF}(f,k) = \max_{g\in\mathbb{Z}[x],\deg(g)\le k(\deg(f)-1)} \|g\|_f \,/\, \|g\|_\infty .$$

For example, a simple calculation shows that $\mathrm{EF}(x^n \pm 1, k) \le k$ and $\mathrm{EF}(x^{n-1} + x^{n-2} + \cdots + 1, k) \le 2k$. We say a polynomial $f$ is suitable if $f$ is a monic and irreducible in $\mathbb{Z}[x]$ and there is a constant $c$ such that $\mathrm{EF}(f,k) \le ck$ for any natural number $k$. The security of the hash functions is based on the worst-case hardness of $\Lambda(f)$-SVP for a suitable polynomial $f$. See [13] for more details. They adopted a family of polynomials such as $x^n + 1$ and $x^{n-1} + x^{n-2} + \cdots + 1$ for $n$ such that the polynomials are irreducible in $\mathbb{Z}[x]$.

To describe the hash functions, we prepare some notations. Define $\mathrm{Rot}_f : \mathbb{Z}_q[x]/\langle f\rangle \to \mathbb{Z}_q^{n\times n}$ as

$$\mathrm{Rot}_f(\mathbf{a}) := [\mathbf{e}_0 \otimes \mathbf{a}, \mathbf{e}_1 \otimes \mathbf{a}, \ldots, \mathbf{e}_{n-1} \otimes \mathbf{a}] ,$$

where each $\mathbf{e}_i$ is a polynomial $x^i$ and the operator $\otimes$ denotes the product operator in $\mathbb{Z}_q[x]/\langle f\rangle$. For example, if $f = x^n - 1$ or $f = x^n + 1$, $\mathrm{Rot}_f(\mathbf{a})$ is a circulant or skew-circulant matrix, respectively. We next define $ROT(f,q,m')$ as a subset of $\mathbb{Z}_q^{n\times m'n}$,

$$ROT(f,q,m') := \left\{\mathbf{A} = [\mathrm{Rot}_f(\mathbf{a}_1) \ \ldots \ \mathrm{Rot}_f(\mathbf{a}_{m'})] \mid \mathbf{a}_1, \ldots, \mathbf{a}_{m'} \in \mathbb{Z}_q^n\right\}.$$

Let $D(m',d) = \{\mathbf{x} \in \mathbb{Z}^{m'n} \mid \|\mathbf{x}\|_\infty \le d\}$.

We now describe a family of hash functions given in [13].

$$\mathcal{H}_{\mathcal{I}}(f,q,m',d) = \left\{h_{\mathbf{A}} : D(m',d) \to \mathbb{Z}_q^n \mid \mathbf{A} \in ROT(f,q,m')\right\},$$

where $h_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$. We note that, for $\mathbf{A} = [\mathrm{Rot}_f(\mathbf{a}_1) \ \ldots \ \mathrm{Rot}_f(\mathbf{a}_{m'})]$ and $\mathbf{x} = \mathbf{x}_1 \circ \ldots \circ \mathbf{x}_{m'}$, we can interpret $\mathbf{A}\mathbf{x}$ into $\sum_{i=1}^{m'} \mathbf{a}_i \otimes \mathbf{x}_i$.

Lyubashevsky and Micciancio showed the following theorem in [13].

**Theorem 5.** *Let $E = \mathrm{EF}(f,3)$. Let $m' > \log q/\log 2d$ and $q > 2Edm'n^{3/2}\log n$. Then for $\gamma = 8E^2dm'n\log^2 n$, if $\Lambda(f)$-$\mathrm{SVP}_\gamma^\infty$ is hard in the worst case then $\mathcal{H}_{\mathcal{I}}(f,q,m',d)$ is collision resistant.*

Next, let $\mathcal{H}'_{\mathcal{I}}(f,q,m')$ be a restricted version of the above hash functions:

$$\mathcal{H}'_{\mathcal{I}}(f,q,m') = \left\{h_{\mathbf{A}} : \{0,1\}^{m'n} \to \mathbb{Z}_q^n \mid \mathbf{A} \in ROT(f,q,m')\right\}.$$

**Corollary 1.** *For any $m'(n) = \Theta(\log n)$, there exists $q(n) = \Theta(m'n^{3/2}\log n)$ and $\gamma = \Theta(mn\log^2 n)$, such that, for a suitable polynomial $f$, $\mathcal{H}'_{\mathcal{I}}(f,q,m')$ is collision resistant if $\Lambda(f)$-$\mathrm{SVP}_\gamma^\infty$ is hard in the worst case.*

Finally, we introduce a restricted version of $\mathcal{H}_{\mathcal{I}}(f,q,m')$ for use in our identification scheme:

$$\mathcal{H}'_{\mathcal{I}}(f,q,m',w) = \left\{h_{\mathbf{A}} : \mathrm{B}(m'n,w) \to \mathbb{Z}_q^n \mid \mathbf{A} \in ROT(f,q,m')\right\}.$$

Now, the setup algorithm on input $1^n$ outputs $\mathbf{A} = [\mathrm{Rot}_f(\mathbf{a}_1) \ \ldots \ \mathrm{Rot}_f(\mathbf{a}_{m'})]$ from $ROT(f,q,m')$ uniformly at random. The key-generation algorithm on input $\mathbf{A}$, chooses a random vector $\mathbf{x} \in \mathrm{B}(m'n,w)$ uniformly at random, computes a vector $\mathbf{y} := \mathbf{A}\mathbf{x}$, and outputs $(pk,sk) = (\mathbf{y},\mathbf{x})$.

### C.2 An String Commitment Scheme

Using $\mathcal{H}'_I(f, q, m')$, we also obtain a simple string commitment scheme. We apply the following lemma to $\mathcal{H}'_I(f, q, m')$ and obtain the statistical-hiding property of a string commitment scheme. This lemma is obtained by use of Micciancio's regularity lemma [14]

**Lemma 3.** *Let $q$ be a prime $q = q(n) = n^{O(1)}$ and $m'$ an integer such that $m' = m'(n) > 2\log q$. Let $f \in \mathbb{Z}_q[x]$ of degree $n$ be a suitable polynomial for $q$. The statistical distance between $(\mathbf{a}_1, \ldots, \mathbf{a}_{m'}, \sum_{i=1}^{m'} \mathbf{a}_i \otimes \mathbf{x}_i)$ and the uniform distribution over the set $(\mathbb{Z}_q[x]/\langle f \rangle)^{m'+1}$ is negligible in $n$.*

Now, we obtain the following lemma as in Lemma 1.

**Lemma 4.** *For any $m'(n) = \Theta(\log n)$, there exists $q(n) = \Theta(m'n^{3/2}\log n)$ and $\gamma = \Theta(m'n\log^2 n)$, such that, $m'(n) > 4\log q$ and for a suitable polynomial $f$, $\mathrm{Com}_{\mathbf{A}}$ is a statistically hiding and computationally binding string commitment scheme in the trusted setup model if $\Lambda(f)$-$\mathrm{SVP}^{\infty}_{\gamma}$ is hard in the worst case.*

Using the Merkle-Damgård technique, we obtain the string commitment scheme whose commitment function is $\mathrm{Com}_{\mathbf{A}} : \{0,1\}^* \times \{0,1\}^{m'n/2} \to \mathbb{Z}_q^n$ rather than $\mathrm{Com}_{\mathbf{A}} : \{0,1\}^{m'n/2} \times \{0,1\}^{m'n/2} \to \mathbb{Z}_q^n$.

### C.3 An Identification Scheme and An Ad Hoc Identification Scheme

We obtain the ID scheme and the AID scheme by combining the above setup and key-generation algorithms and the string commitment scheme with Stern's scheme as in Section 4 and 5. One can prove the securities of the schemes in the same manner to the proof of Theorems 2 and 4.

**Theorem 6.** *Let $f$ be a polynomial and $E := \mathrm{EF}(f, 3)$. Let $m' = m'(n)$, $q = q(n)$, and $w = w(n)$ be polynomially bounded functions such that $m' > 4\log q$, $q > 2Em'n^{3/2}\log n$, and $q^n/|\mathrm{B}(m'n, w)|$ is negligible in $n$. Assume that $f$ is a suitable polynomial. Then for $\gamma = 8E^2m'n\log^2 n$, if $\Lambda(f)$-$\mathrm{SVP}^{\infty}_{\gamma}$ is hard in the worst case then the ID scheme $\mathrm{S}_{\mathrm{C/IL}}$ which uses the above setup and key-generation algorithms and the above string commitment scheme is secure against impersonation under concurrent attacks.*

*Sketch of proof:* We show that if there exists an impersonator $\mathcal{I}$ which succeeds impersonation under concurrent attacks with non-negligible probability $\epsilon$, there exists $\mathcal{A}$ that finds a collision $(\mathbf{z}_1, \mathbf{z}_2)$ for $\mathcal{H}'_I(f, q, m')$.

Given $\mathbf{A} \in ROT(f, q, m')$, $\mathcal{A}$ chooses a random secret key $\mathbf{x} \in \mathrm{B}(m'n, w)$ and compute $\mathbf{y} := \mathbf{A}\mathbf{x}$. $\mathcal{A}$ executes $\mathcal{I}$ on inputs $(\mathbf{A}, \mathbf{y})$. We note that $\mathcal{A}$ can simulate the oracles Conv and Prov, since $\mathcal{A}$ has the secret key $\mathbf{x}$. $\mathcal{A}$ executes $\mathcal{I}$ three times with random challenges and a fixed random tape. Then, $\mathcal{A}$ obtains three transcripts $(Cmt^{(i)}, Ch^{(i)}, Rsp^{(i)}, Dec^{(i)})$ for $i = 1, 2, 3$ as the results of the interactions between $\mathcal{I}$ and $\mathcal{A}$. Note that $Cmt^{(1)} = Cmt^{(2)} = Cmt^{(3)}$ since $\mathcal{A}$ fixes the random tape to work $\mathcal{I}$. By the assumption, with non-negligible probability, $\mathcal{A}$ obtains good transcripts such that $Dec^{(i)} = (Dec_1^{(i)}, \ldots, Dec_n^{(i)})$ are all 1 for every $i$. Then, $\mathcal{A}$ can find $\mathbf{x}'$ from $(\mathbf{A}, \mathbf{y})$

or find $(s,\rho) \neq (s',\rho')$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathrm{Com}_{\mathbf{A}}(s';\rho')$ by using the fact that $Cmt^{(1)} = Cmt^{(2)} = Cmt^{(3)}$. In the former case, we can show that $\mathbf{x}' \neq \mathbf{x}$ with probability at least $1/2$ as in the proof of Theorem 2. $\mathcal{A}$ outputs $(\mathbf{x},\mathbf{x}')$. Since $\mathbf{x},\mathbf{x}' \in \mathrm{B}(m'n,w) \subseteq \{0,1\}^{m'n}$, $\mathcal{A}$ indeed finds a collision for $\mathcal{H}'_{\mathcal{I}}(f,q,m')$. In the latter case, $\mathcal{A}$ computes $\mathbf{z} \neq \mathbf{z}' \in \{0,1\}^{m'n}$ from $(s,\rho)$ and $(s',\rho')$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathbf{Az}$ and $\mathrm{Com}_{\mathbf{A}}(s';\rho') = \mathbf{Az}'$. Thus, $\mathcal{A}$ outputs $(\mathbf{z},\mathbf{z}')$ as a collision for $\mathcal{H}'_{\mathcal{I}}(f,q,m')$. □

**Theorem 7.** *Let $f$ be a polynomial and $E := \mathrm{EF}(f,3)$. Let $m' = m'(n)$, $q = q(n)$, and $w = w(n)$ be polynomially bounded functions such that $m' > 4\log q$, $q > 2E(w+1)m'n^{3/2}\log n$, and $q^n/|\mathrm{B}(m'n,w)|$ is negligible in $n$. Assume that $f$ is a suitable polynomial. Then for $\gamma = 8E^2(w+1)m'n\log^2 n$, if $\Lambda(f)$-$\mathrm{SVP}^{\infty}_{\gamma}$ is hard in the worst case then the AID scheme which uses the above setup and key-generation algorithms and the above string commitment scheme is secure against impersonation under concurrent attacks.*

*Sketch of proof:* We show that if there exists an impersonator $\mathcal{I}$ which succeeds impersonation under concurrent chosen-group attacks with non-negligible probability, there exists $\mathcal{A}$ that finds a collision $(\mathbf{z}_1,\mathbf{z}_2)$ for $\mathcal{H}_{\mathcal{I}}(f,q,m',w+1)$.

The algorithm $\mathcal{A}$, given input $\mathbf{A} \in ROT(f,q,m')$, feeds $\mathbf{A}$ to the impersonator $\mathcal{I}$. In the experiment, the impersonator $\mathcal{I}$ will call Init, Corr, and Prov. If $\mathcal{I}$ calls Init with input $i$, then $\mathcal{A}$ chooses $\mathbf{s}_i \in \mathrm{B}(m,w)$ at random, computes $\mathbf{y}_i := \mathbf{As}_i$, and returns $\mathbf{y}_i$ to $\mathcal{I}$. Note that $\mathcal{A}$ can correctly simulate the oracles Corr and Prov, since $\mathcal{A}$ has the secret key $\mathbf{s}_i$ corresponding to the public key $\mathbf{y}_i$.

At the end of the experiment, $\mathcal{I}$ will impersonate as a group which is specified by the set of public keys $R = (\mathbf{y}_1,\ldots,\mathbf{y}_l)$. Rewinding $\mathcal{I}$ three times, $\mathcal{A}$ obtains $(s,\rho) \neq (s',\rho')$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathrm{Com}_{\mathbf{A}}(s';\rho')$ or a vector $\mathbf{x} = \mathbf{x}_1 \circ \mathbf{x}_2$ such that $[\mathbf{A}\,\mathbf{y}_1\ \ldots\ \mathbf{y}_l]\mathbf{x} = \mathbf{0}$, where $\mathbf{x}_1 \in \{-1,0,1\}^{m'n}$, $\mathbf{x}_2 \in \{-1,0,1\}^l$, and $\mathbf{x} \in \mathrm{B}'(m'n+l,w)$ as in the proof of Theorem 2.

In the former case, $\mathcal{A}$ computes $\mathbf{z} \neq \mathbf{z}' \in \{0,1\}^{m'n}$ such that $\mathrm{Com}_{\mathbf{A}}(s;\rho) = \mathbf{Az}$ and $\mathrm{Com}_{\mathbf{A}}(s';\rho') = \mathbf{Az}'$. Hence, $\mathcal{A}$ outputs $(\mathbf{z},\mathbf{z}')$ as a collision for $\mathcal{H}_{\mathcal{I}}(f,q,m',w+1)$.

In the latter case, we have $\mathbf{Ax}_1 + \sum_{i=1}^{l} x_{2,i}\mathbf{y}_i = \mathbf{0}$, that is, $\mathbf{Ax}_1 + \sum_{i=1}^{l} x_{2,i}\mathbf{As}_i = \mathbf{0}$. Hence, we obtain that $\mathbf{A}(\mathbf{x}_1 + \sum_{i=1}^{l} x_{2,i}\mathbf{s}_i) = \mathbf{0}$. By the same argument as in the proof of Theorem 2, we have that $\mathbf{x}_1 + \sum_i x_{2,i}\mathbf{s}_i \neq \mathbf{0}$ with probability at least $1/2$. Recall that the numbers of $+1$ in $\mathbf{x}$ is $w$ and that of $-1$ in $\mathbf{x}$ is 1. Thus we can split the vector $\mathbf{x}_1 + \sum_i x_{2,i}\mathbf{s}_i$ into two vector $\mathbf{z}_1$ and $\mathbf{z}_2$ such that $\mathbf{x}_1 + \sum_i x_{2,i}\mathbf{s}_i = \mathbf{z}_1 + \mathbf{z}_2$, $\mathbf{Az}_1 = \mathbf{Az}_2$, and $\mathbf{z}_1,\mathbf{z}_2 \in D(m',w+1)$. Hence, $\mathcal{A}$ outputs $(\mathbf{z}_1,\mathbf{z}_2)$ as a collision for $\mathcal{H}_{\mathcal{I}}(f,q,m',w+1)$. □