New Smooth Projective Hashing For Oblivious Transfer

Bing Zeng[a]

[a]*School of Software Engineering, South China University of Technology, Guangzhou, 510006, China*

Abstract

Oblivious transfer is an important tool against malicious cloud server providers. Halevi-Kalai OT, which is based on smooth projective hash(SPH), is a famous and the most efficient framework for 1-out-of-2 oblivious transfer (OT_1^2) against malicious adversaries in plain model. A natural question however, which so far has not been answered, is whether its security level can be improved, i.e., whether it can be made fully-simulatable.

In this paper, we press a new SPH variant, which enables a positive answer to above question. In more details, it even makes fully-simulatable OT_t^n ($n, t \in \mathbb{N}$ and $n > t$) possible. We instantiate this new SPH variant under not only the decisional Diffie-Hellman assumption, the decisional N-th residuosity assumption and the decisional quadratic residuosity assumption as currently existing SPH constructions, but also the learning with errors (LWE) problem. Before this paper, there is a folklore that it is technically difficult to instantiate SPH under the lattice assumption (e.g., LWE). Considering quantum adversaries in the future, lattice-based SPH makes important sense.

Keywords: oblivious transfer, secure multiparty computation, malicious adversaries, smooth projective hashing, learning with errors.

1. Introduction

Oblivious transfer (OT) [1] is a fundamental cryptographic primitive allowing the secure multiparty computation (SMPC) of *any* computable function [2]. t-out-of-n oblivious transfer (OT_t^n) deals with the scenario where a sender holds n private values $m_1, m_2, \ldots, m_n$ and a receiver possesses t private indexes $i_1, i_2, \ldots, i_t$. The receiver expects to get the values $m_{i_1}, m_{i_2}, \ldots, m_{i_t}$ without leaking any information about which ones were chosen. On the other hand, the sender does not want the receiver to know anything but the t values queried about.

OT is an important tool against malicious cloud server providers (CSP). For example, a client first stores n files on a CSP, then he can retrieve t files of them via a OT_t^n protocol. This prevents the possibly malicious CSP from deducing the behaviors of the client. Thus, OT provides a privacy preserving way to access cloud data. Generally speaking, it is used as a building block in a variety of security protocols against malicious CSP: privacy preserving data mining [3], database search [4, 5], oblivious keyword search [6],electronic commerce [7].

Smooth projective hashing (SPH) is introduced by Cramer and Shoup [8] in context of constructing public-key encryption scheme secure against adaptive chosen ciphertext attacks. Based on SPH, Halevi and Kalai generalizes highly efficient protocols of [9, 7] and presents a framework for OT_1^2 against malicious adversaries in plain model. Halevi-Kalai OT is a remarkable work among known protocols [10, 11, 12, 5, 13, 14], because it is most efficient and generally realizable.

However, Halevi-Kalai OT is *non-simulatable*. That is, it can not provide simulation-based security proof. Thus, it is harder to use it as a building block in secure multi-party computation protocols. A concrete attack on non-simulatable OT can be seen in [4]. A natural question however, which so far has not been answered, is whether Halevi-Kalai OT can be made fully-simulatable. Before this paper, there is a folklore that it is technically difficult to instantiate SPH under the lattice assumption [14]. Considering quantum adversaries in the future, lattice-based cryptographic primitive make important sense. Another natural question is whether this folklore is true.

Email address: `zeng.bing.zb@gmail.com` (Bing Zeng)

1.1. Our Contribution

In this paper, we present a new SPH. Using this new SPH, [15] not only present a positive answer to the first question but also present a fully-simulatable framework for general OT_t^n ($n, t \in \mathbb{N}$ and $n > t$) . We tress that OT_t^n has its own interesting when compare it with OT_1^2. First, there are many applications for OT_t^n itself. Second, when an efficient OT_t^n protocol is needed in practice, it is unknown how to construct it from a known OT_1^2 protocol. Please see [15] for more discussions.

We present a negative answer to the second question. Specifically, we instantiate this new SPH variant under not only the decisional Diffie-Hellman assumption, the decisional N-th residuosity assumption and the decisional quadratic residuosity assumption as currently existing SPH constructions, but also the learning with errors (LWE) problem.

We present a useful lemma related to probability ensembles. Loosely speaking, we shows that if probability ensembles X and Y are computationally indistinguishable, then multiple probability ensembles ($X_1, X_2, \cdots, X_n$, $Y_1, Y_2, \cdots, Y_l$) and multiple probability ensembles gained by arbitrarily permutating the former are computationally indistinguishable too, where each X_i (Y_i, respectively) is independent and employs the sample algorithm as X (Y, respectively). Please see Lemma 34 for the details.

1.1.1. Cryptographic Approach

The SPH variant used in [16] was called *verifiably smooth projective hash family*. It deals with two types of instances (smooth and projective) which are computationally indistinguishable due to a property called *hard subset membership*. Nonetheless, another property called *verifiable smoothness* provides a way to verify whether at least one of a two instances is smooth. In the remaining of this paper, we are to denote verifiably smooth projective hash family with hard membership property by VSPH-HM.

For each instance x of each type, there are two kinds of keys (hash keys and projection keys). In addition, every projective instance holds a witness while no smooth instance does so. A hash value is computed from a hash key (i.e., $\mathsf{Hash}(x, hk)$) while a projection value is computed from a projection key and a witness (i.e., $\mathsf{pHash}(x, pk, w)$). For a smooth instance, the projection value reveals almost no knowledge about the hash value due to a property called *smoothness*. However, for a projective instance, the projection value equals the hash value. This fact is guaranteed by another property called *projection*.

Despite the notion of VSPH-HM can be used to deal with OT_1^n, it seems difficult to extend it to handle the general case OT_t^n. The reason is that, to hold verifiable smoothness, both types of instances have to be generated in a dependent way. This makes it difficult to design an algorithm checking that at least t of n arbitrary instances are smooth without leaking any information to the adversaries which could be used to distinguish smooth instances from projective instances. Therefore, even constructing a non-simulatable protocol for OT_t^n as in [16] seems difficult.

Another problem comes from the fact that, for a protocol using a VSPH-HM, it is impossible to gain simulation-based security in the case where only the receiver is corrupted. The reason is that, to extract the adversary's real input in this case, the simulator has to identify the projective part of a smooth-projective instance pair. However, this is computationally impossible because of the hard subset membership assumption.

Seeing the above difficulties, we define a new variant of SPH called *smooth projective hash family with distinguishability and hard subset membership* (SPH-DHM). The most essential difference between the notions of SPH-DHM and VSPH-HM is that a SPH-DHM also provides witnesses to the smooth instances while verifiable smoothness is removed. Furthermore, we introduce the *distinguishability* property providing a way to differentiate smooth instances from projective ones when needed witnesses are given. This enables a SPH-DHM to generate both types of instances independently. Further, this enables a SPH-DHM to treat fully-simulatable OT_t^n. Please see [15] for more discussions.

1.1.2. Instantiation of the SPH

We will present an instantiation of a SPH-DHM based on the learning with errors (LWE) assumption following the original design of the public key cryptosystem from [17]. Prior to our construction, it was thought to be technically difficult to instantiate a SPH under this assumption. We observed that the algorithm $\mathsf{Hash}(.)$ (computing hash values) could be viewed as an encryption algorithm while $\mathsf{pHash}(.)$ (computing projection values) could be interpreted as a decryption algorithm. Our instantiation idea is to take public-private key pairs as projective instance-witness pairs and take messy public-private key pairs[1] as smooth instance-witness pairs. With this identification, we can see that normal public keys and messy public keys can provide projection and smoothness, respectively. To implement this idea, we let the key generator (i.e., the algorithm generating the hash-projection key pairs) take an instance as a part of its input. This is a technical difference between a SPH-DHM and a SPH. Besides the LWE assumption, we also show that a SPH-DHM can be instantiated under the DDH assumption, the decisional N-th residuosity (DNR) assumption and the decisional quadratic residuosity (DQR) assumption as for previous existing SPH.

1.2. Related Work

Smooth projective hashing is introduced by Cramer and Shoup [8]. It usually provides conceptual simplicity, modular design, and generally realizability. Originally it is used to construct efficient public-key encryption scheme (PKE) secure against adaptive chosen ciphertext attacks. Lately SPH is widely used in various contexts such as OT and password-based authenticated key exchange (PAKE).

OT [1] is a fundamental cryptographic primitive allowing the secure multiparty computation (SMPC) of *any* computable function [2]. Based on SPH, Halevi and Kalai present a framework for OT^2_1 against malicious adversaries in plain model [16], which generalizes the protocols of [9, 7]. [18] extends Halevi-Kalai OT and present a framework for OT^n_t against covert adversaries. There are many SPH-based works achieve higher security level, i.e., universally composable security against adaptive adversaries, which additionally require a trusted setup assumption, i.e., a trusted common reference string. For example, [19] gives a OT^2_1 scheme and [20, 21] proposes generic constructions of OT^n_1.

PKE. [8] present and use SPH to generalize their previous PKE. Later [22] finds that SPH can be applied to the leakage-resilient PKE. Following works can be seen in [23, 24]. In constructing identity-based encryption schemes against key leakage attacks, Alwen et al. [25] firstly give the concept of ID-based SPH and get three encryption schemes. Recently [26] employs SPH to present a framework for PKE satisfying key-dependent message security and a framework for dual-mode cryptosystems.

PAKE is proposed by Bellovin and Merritt [27] where authentication is done using a simple password, possibly drawn from a small entropy space subject to exhaustive search. SPH has been extensively used in PAKE, starting with the work of Gennaro and Lindell [28] which generalized an earlier construction by [29]. Following Gennaro-Lindell methodology, [30] shows a UC secure PAKE protocol and [31, 20] presents an adaptively secure PAKE. Later new variants of SPHF are proposed to enable the construction of one-round PAKE schemes [32, 33, 21].

1.3. Paper Organization

In the next section, we describe the notations used throughout our work and the security definition for OT^n_t. In Section 3, we define a new variant of smooth projective hash (i.e. SPH-DHM). In Section 4, we reduce the construction of SPH-DHM to considerably simpler hash families. In Section 5, we instantiate those simpler families under the DDH, LWE, DQR, DNR assumptions, respectively.

2. Preliminaries

Most notations and concepts mentioned in this section come from [34, 35, 36] and we tailored them to deal with OT^n_t.

2.1. Basic Notations and Definitions

We set the following notations for this paper:

- $\mathbb{N}$: set of natural numbers.
- k: *security parameter* where $k \in \mathbb{N}$. It is used to measure the security of the underlying computational assumptions (e.g., the DDH assumption).
- $[n]$: the set $\{1, 2, \ldots, n\}$ where $n \in \mathbb{N}$.

[1]A key is called *messy* if a ciphertext generated under such a key carries no information (statistically) about the corresponding plaintext.

- $\Psi = \{T \subseteq [n] : |T| = t\}$: the set of the receiver's all legal private inputs in t-out-of-n oblivious transfer.
- $\vec{x}\langle j\rangle$: the j-th entry of the vector $\vec{x}$.
- S_n: the set of all permutations of $[n]$ where $n \in \mathbb{N}$.
- $\sigma(\vec{x})$: the vector gained by shifting the i-th entry of the n-vector $\vec{x}$ to the $\sigma(i)$-th entry where $\sigma \in S_n$. In other words, $\sigma(\vec{x})$ denotes the vector $\vec{y}$ such that: $\forall i \in [n] \quad \vec{y}\langle\sigma(i)\rangle = \vec{x}\langle i\rangle$.
- $\mathsf{Poly}(.)$: an unspecified positive polynomial.
- $\{0,1\}^*$: set of all bitstrings.
- $\alpha \in_U D$: an element α chosen uniformly at random from a domain D.
- $\alpha \in_\chi D$: an element α chosen from a domain D according the probabilistic distribution χ.
- $|X|$: the cardinality of a finite set X.

Definition 1. *A (positive) function $\mu(.)$ is called* negligible in *k, if and only if:*

$$\forall \mathsf{Poly}(\cdot) > 0\, \exists k_0 \in \mathbb{N} : \forall k > k_0 \quad \mu(k) < 1/\mathsf{Poly}(k).$$

Definition 2. *A* probability ensemble

$$X \overset{def}{=} \{X(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*}$$

is an infinite sequence of random variables indexed by (k, a), where a represents various types of inputs used to sample the instances according to the distribution of the random variable $X(1^k, a)$.

Definition 3. *A probability ensemble X is* polynomial-time constructible*, if there exists a* probabilistic polynomial-time *(PPT) sampling algorithm* $\mathsf{Samp}(.)$ *such that for any a, any k, the random variables* $\mathsf{Samp}(1^k, a)$ *and $X(1^k, a)$ are identically distributed.*

Definition 4. *Let X, Y be two probability ensembles. We say they are* computationally indistinguishable*, denoted by $X \overset{c}{=} Y$, if for any non-uniform PPT algorithm D with auxiliary input $z = (z_k)_{k\in\mathbb{N}}$ (where each $z_k \in \{0,1\}^*$), there exists a negligible function $\mu(.)$ such that for any sufficiently large k and any $a \in \{0,1\}^*$, it holds that*

$$|\mathrm{Prob}(D(1^k, a, X(1^k, a), z_k) = 1) - \mathrm{Prob}(D(1^k, a, Y(1^k, a), z_k) = 1)| \leqslant \mu(k).$$

Definition 5. *Let X, Y be two probability ensembles. They are said to be* statistically indistinguishable*, denoted by $X \overset{s}{=} Y$, if their statistical difference is negligible. More specifically, if there exists a negligible function $\mu(.)$ such that*

$$1/2 \cdot \sum_{\alpha\in\{0,1\}^*} |\mathrm{Prob}(X(1^k, a) = \alpha) - \mathrm{Prob}(Y(1^k, a) = \alpha)| = \mu(k).$$

Definition 6. *Let X, Y be two probability ensembles. They are said to be* identical*, denoted by $X \equiv Y$, if the distributions of $X(1^k, a)$ and $Y(1^k, a)$ are identical.*

Remark 7. *Let X, Y be two probability ensembles. We obviously have: $X \equiv Y$ implies $X \overset{s}{=} Y$ and $X \overset{s}{=} Y$ implies $X \overset{c}{=} Y$.*

2.2. *Smooth Projective Hash*

Halevi and Tauman Kalai applied the following SPH variant to construct a protocol for OT^2_1.

Definition 8 ([16]). *A hash family $\mathcal{H}$ is defined by means of the following PPT algorithms $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{KG}, \mathsf{Hash}, \mathsf{pHash})$:*

- *Parameter generator* PG*: it takes a security parameter k as input and returns a hash parameter Λ: i.e. $\Lambda \leftarrow \mathsf{PG}(1^k)$.*
- *Instance sampler* IS*: it takes a security parameter k and a hash parameter Λ as input and returns a triple, i.e., $(\dot{x}, \dot{w}, \ddot{x}) \leftarrow \mathsf{IS}(1^k, \Lambda)$), where $\dot{x}$ is a projective instance, $\dot{w}$ is one of its witnesses, $\ddot{x}$ is a smooth instance.*
- *Instance-testing algorithm* IT*: it tests the parameters Λ and two strings x_0, x_1, i.e., $\mathsf{IT}(\Lambda, x_0, x_1) \in \{0, 1\}$. The intent is to test that at least one of x_0, x_1 is a smooth instance.*
- *Key generator* KG*: it takes a security parameter k, a hash parameter Λ and an instance x as input and outputs a hash-projection key pair (hk, pk): i.e., $(hk, pk) \leftarrow \mathsf{KG}(1^k, \Lambda)$.*
- *Hash algorithm* Hash*: it takes a security parameter k, a hash parameter Λ, an instance x and a hash key hk as input and outputs a value y: i.e., $y \leftarrow \mathsf{Hash}(1^k, \Lambda, x, hk)$.*
- *Projection algorithm* pHash*: it takes a security parameter k, a hash parameter Λ, an instance x, a projection key pk and a witness w of x as input and outputs a value y: i.e., $y \leftarrow \mathsf{pHash}(1^k, \Lambda, x, pk, w)$.*

The *smoothness* requires that for any $\ddot{x}$, its projection key and hash value are almost uniformly distributed. The *projection* requires that for any $\dot{x}$ and any its hash-projection key pair (hk, pk), its hash value equals its projection value. The *verifiable smoothness* requires that if $\mathsf{IT}(\Lambda, x_0, x_1) = 1$, then at least one of x_0, x_1 is a smooth instance. The *hard subset membership* requires the smooth instances $\ddot{x}$ and projective instances $\dot{x}$ are computationally indistinguishable. We let VSPH-HM denote the hash family which holds all properties mentioned here.

3. A New Smooth Projective Hash

Since previous definitions of SPH do not suffice for our application, we define another variant of SPH.

Definition 9. *A (n, t)-hash family $\mathcal{H}$ is defined by means of the following PPT algorithms $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{pIS}, \mathsf{Check}, \mathsf{DI}, \mathsf{KG}, \mathsf{Hash}, \mathsf{pHash})$:*

- *Parameter generator* PG*: it takes a security parameter k as input and returns a hash parameter Λ: i.e. $\Lambda \leftarrow \mathsf{PG}(1^k)$.*
- *Checker* Check*: it takes a security parameter k and a hash parameter Λ as input and returns an indicator bit $b \in \{0, 1\}$: i.e. $b \leftarrow \mathsf{Check}(1^k, \Lambda)$. The objective is to check that Λ was correctly generated.*
- *Instance sampler* IS*: it takes a security parameter k and a hash parameter Λ as input and returns a vector $\vec{a} = ((\dot{x}_1, \dot{w}_1), \ldots, (\dot{x}_t, \dot{w}_t), (\ddot{x}_{t+1}, \ddot{w}_{t+1}), \ldots, (\ddot{x}_n, \ddot{w}_n))$ (i.e., $\vec{a} \leftarrow \mathsf{IS}(1^k, \Lambda)$) where each entry of $\vec{a}$ is an instance-witness pair with the first t pairs are projective and the last $n - t$ pairs are smooth.*
- *Projective instance sampler* pIS*: similar to* IS *with exception that all n instances are projective.*
- *Distinguisher* DI*: it takes a security parameter k, a hash parameter Λ and an instance-witness pair (x, w) as input and outputs an indicator value b: i.e., $b \leftarrow \mathsf{DI}(1^k, \Lambda, x, w)$. Its goal is to distinguish smooth instances and projective instances.*
- *Key generator* KG*: it takes a security parameter k, a hash parameter Λ and an instance x as input and outputs a hash-projection key pair (hk, pk): i.e., $(hk, pk) \leftarrow \mathsf{KG}(1^k, \Lambda, x)$.*
- *Hash algorithm* Hash*: it takes a security parameter k, a hash parameter Λ, an instance x and a hash key hk as input and outputs a value y: i.e., $y \leftarrow \mathsf{Hash}(1^k, \Lambda, x, hk)$.*

- *Projection algorithm* pHash*: it takes a security parameter k, a hash parameter Λ, an instance x, a projection key pk and a witness w of x as input and outputs a value y: i.e., $y \leftarrow \mathsf{pHash}(1^k, \Lambda, x, pk, w)$.*

Definition 10. *For a given hash parameter Λ, if* Check *outputs* 1*, then Λ is said to be* legal*; otherwise, it is said to be* illegal.

Remark 11. *It is obvious that any Λ generated by* PG *is legal.*

Definition 12. *Let $R = \{(x, w) : x, w \in \{0, 1\}^*\}$ be a relation. For a legal Λ, we define its* projective relation *as $\dot{R}_\Lambda = \{(\dot{x}, \dot{w}) : (\dot{x}, \dot{w})$ is generated by $\mathsf{IS}(1^k, \Lambda)\}$ and its* smooth relation *as $\ddot{R}_\Lambda = \{(\ddot{x}, \ddot{w}) : (\ddot{x}, \ddot{w})$ is generated by $\mathsf{IS}(1^k, \Lambda)\}$.*

Definition 13 (Distinguishability)**.** *For any legal hash parameter Λ, any instance-witness pair (x, w), we require:*

$$\mathsf{DI}(1^k, \Lambda, x, w) = \begin{cases} 0 & \textit{if } (x, w) \in \dot{R}_\Lambda, \\ 1 & \textit{if } (x, w) \in \ddot{R}_\Lambda, \\ 2 & \textit{otherwise.} \end{cases}$$

Definition 14. *If R is a relation, then its* language *is defined as $L \overset{def}{=} \{x \in \{0, 1\}^* : \exists w((x, w) \in R)\}$.*

Let $\dot{L}_\Lambda$ and $\ddot{L}_\Lambda$ be the languages of relation $\dot{R}_\Lambda$ and relation $\ddot{R}_\Lambda$, respectively. The properties smoothness and projection to be defined next will ensure that $\dot{L}_\Lambda \cap \ddot{L}_\Lambda = \emptyset$ holds. That is, no instance can exhibit both smoothness and projection.

Definition 15 (Projection)**.** *For any hash parameter Λ generated by $\mathsf{PG}(1^k)$, any projective instance-witness pair $(\dot{x}, \dot{w})$ generated by $\mathsf{IS}(1^k, \Lambda)$, and any hash-projection key pair (hk, pk) generated by $\mathsf{KG}(1^k, \Lambda, \dot{x})$, it holds that*

$$\mathsf{Hash}(1^k, \Lambda, \dot{x}, hk) = \mathsf{pHash}(1^k, \Lambda, \dot{x}, pk, \dot{w}).$$

Definition 16. *For an instance-witness vector $\vec{a} = ((x_1, w_1), \ldots, (x_n, w_n))$, we define its* instance vector *as $x(\vec{a}) \overset{def}{=} (x_1, \ldots, x_n)$ and its* witness vector *as $w(\vec{a}) \overset{def}{=} (w_1, \ldots, w_n)$.*

Definition 17. *Fix a legal hash parameter Λ. If a vector $\vec{a}$ contain at least $n - t$ smooth instance-witness pairs, (i.e., at least $n - t$ pairs in $\ddot{R}_\Lambda$), then $\vec{a}$ is said to be* legal.

Remark 18. *Note that any $\vec{a}$ generated by $\mathsf{IS}(1^k, \Lambda)$ is legal, and the legality of any $\vec{a}$ that may be maliciously generated can be checked by invoking algorithm* DI *at most n times.*

Definition 19 (Smoothness)**.** *For any legal hash parameter Λ, any legal instance-witness vector $\vec{a}$ (without loss of generality, we assume that the last $n - t$ entries of $\vec{a}$ are smooth), any permutation $\sigma \in S_n$,* smoothness *holds if the two probability ensembles $\mathsf{SM}_1 \overset{def}{=} \{\mathsf{SM}_1(1^k)\}_{k \in \mathbb{N}}$ and $\mathsf{SM}_2 \overset{def}{=} \{\mathsf{SM}_2(1^k)\}_{k \in \mathbb{N}}$, specified as follows, are statistically indistinguishable: i.e., $\mathsf{SM}_1 \overset{s}{=} \mathsf{SM}_2$.* Perfect smoothness *holds, if $\mathsf{SM}_1 \equiv \mathsf{SM}_2$.*

- *$G_\Lambda \overset{def}{=} \{y : x \in \dot{L}_\Lambda \cup \ddot{L}_\Lambda, (hk, pk) \leftarrow \mathsf{KG}(1^k, \Lambda, x), y \leftarrow \mathsf{Hash}(1^k, \Lambda, x, hk)\}$ is a set of all possible hash values.*
- *Algorithm $\mathsf{SmGen}_1(1^k)$ works as follows:*
  - *$\vec{x} \leftarrow x(\vec{a})$.*
  - *For each $j \in [n]$, perform: $(hk_j, pk_j) \leftarrow \mathsf{KG}(1^k, \Lambda, \vec{x}\langle j \rangle)$, $y_j \leftarrow \mathsf{Hash}(1^k, \Lambda, \vec{x}\langle j \rangle, hk_j)$.*
  - *Set $\overrightarrow{pky} \leftarrow (pk_j, y_j)_{j \in [n]}$ and output $\overrightarrow{pky}$.*
- *Algorithm $\mathsf{SmGen}_2(1^k)$ works as $\mathsf{SmGen}_1(1^k)$ except that for each $j \in \{t + 1, t + 2, \ldots, n\}$, $y_j \in_U G_\Lambda$.*
- *For $i \in [2]$, algorithm $\mathsf{SM}_i(1^k)$ works as follows: $\overrightarrow{pky} \leftarrow SmGen_i(1^k)$, $\overrightarrow{\widetilde{pky}} \leftarrow \sigma(\overrightarrow{pky})$ and output $\overrightarrow{\widetilde{pky}}$.*

Definition 20 (Hard Subset Membership). *For any $\sigma \in S_n$, the two probability ensembles* $\mathsf{HS}_1 \overset{def}{=} \{\mathsf{HS}_1(1^k)\}_{k\in\mathbb{N}}$ *and* $\mathsf{HS}_2 \overset{def}{=} \{\mathsf{HS}_2(1^k)\}_{k\in\mathbb{N}}$*, specified as follows, are computationally indistinguishable, i.e.,* $\mathsf{HS}_1 \overset{c}{=} \mathsf{HS}_2$.

- *Algorithm* $\mathsf{HS}_1(1^k)$ *works as follows:* $\Lambda \leftarrow \mathsf{PG}(1^k)$, $\vec{a} \leftarrow \mathsf{IS}(1^k, \Lambda)$ *and outputs* $(\Lambda, x(\vec{a}))$.
- *Algorithm* $\mathsf{HS}_2(1^k)$ *operates as* $\mathsf{HS}_1(1^k)$ *except that it outputs* $(\Lambda, \sigma(x(\vec{a})))$.

Remark 21. *In this paper, for a projective instance $\dot{x}$, its witness $\dot{w}$ is mainly used to gain the instance's hash value while, for a smooth instance $\ddot{x}$, its witness $\ddot{w}$ serves as a proof of smoothness (i.e., a proof of $\ddot{x} \in \ddot{L}_\Lambda$). The distinguishability property guarantees that, given the needed witness-vector, projective instances and smooth instances are distinguishable.*

For notational simplicity, we denote a (n,t)-hash family $\mathcal{H}$ that holds properties smoothness, projection, distinguishability and hard subset membership (n,t)-SPH-DHM. Similarly, we denote a verifiably smooth projective hash family with hard subset membership property [16] VSPH-HM. As both our SPH-DHM notion and Halevi and Tauman Kalai's VSPH-HM are used to construct protocols for OT, it is necessary to discuss the differences between these two variants of SPH.

1. The major difference between these two notions is that a SPH-DHM not only provides a witness to each projective instance (as a VSPH-HM) but also to every smooth instance. For example, $(2,1)$-SPH-DHM samples tuples of form $((\dot{x}_1, \dot{w}_1), (\ddot{x}_2, \ddot{w}_2))$ while VSPH-HM samples tuples of form $(\dot{w}, \dot{x}, \ddot{x})$.
2. The hard subset membership property of SPH-DHM is stronger than that of VSPH-HM, because the former requires that indistinguishability holds for multiple instances.
3. The key generation algorithm KG of a SPH-DHM takes an additional parameter: an instance x. This technical modification makes instantiating such a hash family easier as we will see in Section 5 using the LWE assumption.
4. The instance sampling algorithm IS of a VSPH-HM generates tuples consisting of a smooth instance, a projective instance and its witness. To deal with OT^n_t, in a SPH-DHM, the sampling algorithm returns vectors containing t projective instance-witness pairs and $n-t$ smooth instance-witness pairs. As a natural result, the properties of smoothness and hard subset membership are extended to consider instance vectors of n entries.
5. A SPH-DHM does not need the verifiable smoothness property of a VSPH-HM (implemented by algorithm IT in [16]). This property was used by Halevi and Tauman Kalai to verify whether at least one of two instances is smooth. Instead, a SPH-DHM exhibits the distinguishability property (implemented by algorithm DI).
6. A SPH-DHM additionally provides an algorithm pIS which plays a key role in simulation-based security proof. In the case that the sender is corrupted, our simulator invokes pIS to cheat the adversary and extract its real input. However, Halevi-Tauman can not offer a simulation-based proof in this case.

4. Reducing the Construction Conditions of a SPH-DHM

In order to obtain a (n,t)-SPH-DHM, we need to satisfy a large number of properties and consider multiple instances. In this section, we reduce the design of such a family to the construction of considerably simpler hash families. The basic idea is to generate the instances of each type (projective/smooth) independently.

4.1. Basic Hash Family

Definition 22. A basic hash family $\mathcal{H}$ *is defined by means of the same seven PPT algorithms* $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{Check}, \mathsf{DI}, \mathsf{KG}, \mathsf{Hash}, \mathsf{pHash})$ *as in Definition 9 with the exception of algorithm* IS *is defined as follows.*

- Instance sampler IS*: it takes a security parameter k, a hash parameter* Λ*, a* work mode $\delta \in \{0,1\}$ *as input and outputs an instance-witness pair* (x, w) *(i.e.,* $(x, w) \leftarrow \mathsf{IS}(1^k, \Lambda, \delta)$*). Its goal is to generate projective instance-witness pairs under mode* 0 *and smooth ones under mode* 1.

To be consistent with Definition 9, we write the instance-witness pair generated under mode $\delta = 0$ as $(\dot{x}, \dot{w})$, and under mode $\delta = 1$ as $(\ddot{x}, \ddot{w})$. The properties of distinguishability and projection are defined as in Section 3. However, the properties of smoothness and hard subset membership are defined in considerably simpler way.

Definition 23 (Smoothness)**.** *For any legal hash parameter Λ, any instance-witness $(\ddot{x}, \ddot{w})$ generated by* $\mathsf{IS}(1^k, \Lambda, 1)$, smoothness *holds if the two probability ensembles* $\mathsf{BSM}_1 \overset{def}{=} \{\mathsf{BSM}_1(1^k)\}_{k\in\mathbb{N}}$ *and* $\mathsf{BSM}_2 \overset{def}{=} \{\mathsf{BSM}_2(1^k)\}_{k\in\mathbb{N}}$*, specified as follows, are statistically indistinguishable, i.e.,* $\mathsf{BSM}_1 \overset{s}{=} \mathsf{BSM}_2$. Perfect smoothness *holds, if* $\mathsf{BSM}_1 \equiv \mathsf{BSM}_2$.

- $\mathsf{BSM}_1(1^k)$*: Generate $(hk, pk) \leftarrow$* $\mathsf{KG}(1^k, \Lambda, \ddot{x})$ *and compute $y \leftarrow$* $\mathsf{Hash}(1^k, \Lambda, \ddot{x}, hk)$*. Output (pk, y).*
- $\mathsf{BSM}_2(1^k)$*: Compared to* $\mathsf{BSM}_1(1^k)$*, the only difference is that $y \in_U G_\Lambda$ where G_Λ is a set of all possible hash values (see Definition 19).*

Definition 24 ([8, 16])**.** *Let $0 < \varepsilon < 1$. For any legal hash parameter Λ, any instance-witness $(\ddot{x}, \ddot{w})$ generated by* $\mathsf{IS}(1^k, \Lambda, 1)$*,* ε-universality *holds if for any string $pk \in \{0, 1\}^*$, and any value $y \in G_\Lambda$, it holds that*

$$\mathrm{Prob}(\mathsf{Hash}(1^k, \Lambda, \ddot{x}, HK) = y \,|\, PK = pk) \leq \varepsilon,$$

where $(HK, PK) \leftarrow$ $\mathsf{KG}(1^k, \Lambda, \ddot{x})$*. The probability is taken over the randomness used by* KG.

Intuitively speaking, universality requires that, for any instance-witness $(\ddot{x}, \ddot{w})$ generated by $\mathsf{IS}(1^k, \Lambda, 1)$, the probability of guessing its hash value is at most ε. Compared with smoothness, ε-universality relaxes the upper bound of this probability to be a higher value. However, we can efficiently gain smoothness from universality.

Lemma 25 ([8, 16])**.** *Given a basic hash family $\widetilde{\mathcal{H}}$ holding universality, then we can efficiently construct a basic hash family $\mathcal{H}$ holding smoothness.*

The constructive proof of this lemma proceeds in two steps as follows. First, it reduces ε-universality to ε^c-universality by simple c-fold "parallelization". Second, smoothness is obtained from ε^c-university by simply applying the Leftover Hash Lemma (see [37] for this lemma). We refer the reader to [8, 16] for the proof details.

Definition 26 (Hard subset membership)**.** *The two probability ensembles* $\mathsf{BHS}_1 \overset{def}{=} \{\mathsf{BHS}_1(1^k)\}_{k\in\mathbb{N}}$ *and* $\mathsf{BHS}_2 \overset{def}{=} \{\mathsf{BHS}_2(1^k)\}_{k\in\mathbb{N}}$ *defined as follows, are computationally indistinguishable (i.e.,* $\mathsf{BHS}_1 \overset{c}{=} \mathsf{BHS}_2$*).*

- $\mathsf{BHS}_1(1^k)$*: Generate $\Lambda \leftarrow$* $\mathsf{PG}(1^k)$ *and compute $(\dot{x}, \dot{w}) \leftarrow$* $\mathsf{IS}(1^k, \Lambda, 0)$*. Output $(\Lambda, \dot{x})$.*
- $\mathsf{BHS}_2(1^k)$*: Generate $\Lambda \leftarrow$* $\mathsf{PG}(1^k)$ *and compute $(\ddot{x}, \ddot{w}) \leftarrow$* $\mathsf{IS}(1^k, \Lambda, 1)$*. Output $(\Lambda, \ddot{x})$.*

4.2. The Reduction Process

Construction 1 Reduction Procedure

Input: A basic hash family $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{Check}, \mathsf{DI}, \mathsf{KG}, \mathsf{Hash}, \mathsf{pHash})$.

We construct a hash family $\overline{\mathcal{H}}$ (as defined in Definition 9) having the same algorithms $\mathcal{H}$ with exceptions of the instance sampling algorithms $\overline{\mathsf{IS}}$ and $\overline{\mathsf{pIS}}(1^k, \Lambda)$ which are defined as follows.

- $\overline{\mathsf{IS}}(1^k, \Lambda)$: For each $i \in [t]$, $\vec{d}\langle i\rangle \leftarrow \mathsf{IS}(1^k, \Lambda, 0)$. For each $i \in [n] \setminus [t]$, $\vec{d}\langle i\rangle \leftarrow \mathsf{IS}(1^k, \Lambda, 1)$. Return $\vec{d}$.
- $\overline{\mathsf{pIS}}(1^k, \Lambda)$: For each $i \in [n]$, $\vec{d}\langle i\rangle \leftarrow \mathsf{IS}(1^k, \Lambda, 0)$.

Theorem 27. *In Construction 1, if the basic hash family $\mathcal{H}$ has (perfect) smoothness, then $\overline{\mathcal{H}}$ holds this property as well.*

Theorem 28. *In Construction 1, if the basic hash family $\mathcal{H}$ has the property hard subset membership property, then the hash family $\overline{\mathcal{H}}$ holds this property as well.*

We will prove the above two theorems in Section 4.2.1 and Section4.2.2 respectively. A consequence of these two theorems and Lemma 25, we can reduce the construction of a (n, t)-SPH-DHM to the design of a basic hash family having smoothness (or universality), projection, distinguishability and hard subset membership.

4.2.1. Proof of Theorem 27

Before starting our demonstration of Theorem 27, we need some preliminary results.

Definition 29. *For any probability ensemble* $X = \{X(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*}$, *we define its* multiple-sampled probability ensemble $\vec{X}$ *as follows:*

$$\vec{X} = \{\vec{X}(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*} \text{ with}$$
$$\vec{X}(1^k, a) = (X_1(1^k, a), X_2(1^k, a), \ldots, X_n(1^k, a)),$$

where each $X_i(1^k, a) \equiv X(1^k, a)$ *and each random variable* $X_i(1^k, a)$ *is independent.*

Lemma 30 ([34]). *Let* X, Y *be two probability ensembles, and* $\vec{X}$, $\vec{Y}$ *their multiple-sampled probability ensembles, respectively.*

- *If* $X \overset{s}{=} Y$, *then* $\vec{X} \overset{s}{=} \vec{Y}$.
- *If* X *and* Y *are polynomial-time constructible and* $X \overset{c}{=} Y$, *then* $\vec{X} \overset{c}{=} \vec{Y}$.

Definition 31. *Let* $F \overset{def}{=} (f_k)_{k\in\mathbb{N}}$ *be a function family. For a probability ensemble* $X = \{X(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*}$, *we define its* image probability ensemble $F(X)$ *under* F *as follows:*

$$F(X) = \{f_k(X(1^k, a))\}_{k\in\mathbb{N}, a\in\{0,1\}^*}.$$

Lemma 32. *Let* X, Y *be two probability ensembles and* $F(X)$, $F(Y)$ *their image probability ensembles, respectively. If* $X \overset{s}{=} Y$, *then* $F(X) \overset{s}{=} F(Y)$.

Proof.

$$\frac{1}{2}\sum_{\alpha\in\{0,1\}^*} |\text{Prob}[f_k(X(1^k, a)) = f_k(\alpha)] - \text{Prob}[f_k(Y(1^k, a)) = f_k(\alpha)]| \leq \frac{1}{2}\sum_{\alpha\in\{0,1\}^*} |\text{Prob}[X(1^k, a) = \alpha] - \text{Prob}[Y(1^k, a) = \alpha]|.$$

This shows that the statistical difference between $F(X)$ and $F(Y)$ is less than that between X and Y. □

We can now expose our proof of Theorem 27.

Proof. We only consider smoothness as the proof in the case of perfect smoothness is similar and simpler. We reuse notations $\mathsf{SmGen}_i(1^k)$, $\underline{\mathsf{SM}}_i$ from Definition 19 and BSM_i from Definition 23.

For the hash family $\mathcal{H}$ in Construction 1, we define the following probability ensemble for each $i \in [2]$ and $j \in [n]$.

$$\mathsf{SM}_i^j = \{\mathsf{SM}_i^j(1^k)\}_{k\in\mathbb{N}} \overset{def}{=} \{\mathsf{SmGen}_i(1^k)\langle j\rangle\}_{k\in\mathbb{N}}.$$

It is easy to see that: $\forall j \in \{t+1, t+2, \cdots, n\}$ $\mathsf{SM}_i^j \equiv \mathsf{BSM}_i$. Combining the fact that basic hash family $\mathcal{H}$ holds smoothness (i.e., $\mathsf{BSM}_1 \overset{s}{=} \mathsf{BSM}_2$) and Lemma 30, we have:

$$\{(\mathsf{SM}_1^{t+1}(1^k), \ldots, \mathsf{SM}_1^n(1^k))\}_{k\in\mathbb{N}} \overset{s}{=} \{(\mathsf{SM}_2^{t+1}(1^k), \ldots, \mathsf{SM}_2^n(1^k))\}_{k\in\mathbb{N}}.$$

We introduce the following notations: $\vec{X} \overset{def}{=} \{(\mathsf{SM}_1^1(1^k), \ldots, \mathsf{SM}_1^n(1^k))\}_{k\in\mathbb{N}}$ and $\vec{Y} \overset{def}{=} \{(\mathsf{SM}_2^1(1^k), \ldots, \mathsf{SM}_2^n(1^k))\}_{k\in\mathbb{N}}$. Looking at the definition of $\mathsf{SmGen}_i(1^k)$, we notice: $\forall j \in [t]$ $\mathsf{SM}_1^j(1^k) \equiv \mathsf{SM}_2^j(1^k)$. So, it holds: $\vec{X} \overset{s}{=} \vec{Y}$.

Let $F \overset{def}{=} (\sigma)_{k\in\mathbb{N}}$ where σ is an arbitrary permutation over set $[n]$. Following Lemma 32, we have $F(\vec{X}) \overset{s}{=} F(\vec{Y})$, i.e.:

$$\{\sigma(\mathsf{SM}_1^1(1^k), \ldots, \mathsf{SM}_1^n(1^k))\}_{k\in\mathbb{N}} \overset{s}{=} \{\sigma(\mathsf{SM}_2^1(1^k), \ldots, \mathsf{SM}_2^n(1^k))\}_{k\in\mathbb{N}}.$$

This means: $\mathsf{SM}_1 \overset{s}{=} \mathsf{SM}_2$. □

4.2.2. Proof of Theorem 28

We start by some preliminary results to be used in our main proof.

Definition 33. *For any probability ensemble* $X = \{X(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*}$ *and* $Y=\{Y(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*}$, *we define their* hybrid probability ensemble $\overrightarrow{XY}$ *as follows:*

$$\overrightarrow{XY}=\{\overrightarrow{XY}(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*} \text{ with } \overrightarrow{XY}(1^k, a)=(X_1(1^k, a), \ldots, X_t(1^k, a), Y_{t+1}(1^k, a), \ldots, Y_n(1^k, a)),$$

where each $X_i(1^k, a) \equiv X(1^k, a)$, *each* $Y_i(1^k, a) \equiv Y(1^k, a)$, *and all random variables are independent.*

Lemma 34. *Let X, Y be two probability ensembles and* $\overrightarrow{XY}$ *their hybrid probability ensemble. Let* $\sigma(\overrightarrow{XY})$ *be the image probability ensemble of* $\overrightarrow{XY}$ *under the function family* $F=(\sigma)_{k\in\mathbb{N}}$ *where* σ *is an arbitrary element of* S_n. *If* $X \overset{c}{=} Y$ *and X, Y are polynomial-time constructible, then* $\overrightarrow{XY} \overset{c}{=} \sigma(\overrightarrow{XY})$.

Proof. If $\{\sigma\langle i\rangle\}_{i\in[t]} \subseteq [t]$, then the relation $\overrightarrow{XY} \overset{c}{=} \sigma(\overrightarrow{XY})$ holds. It remains to consider the case where $\{\sigma\langle i\rangle\}_{i\in[t]} \nsubseteq [t]$.

Assume that the lemma is false in this case. Then, there exists a non-uniform PPT distinguisher D with auxiliary input $z = (z_k)_{k\in\mathbb{N}}$, a polynomial $\mathsf{Poly}(.)$, an infinite positive integer set $G \subseteq \mathbb{N}$ such that, for each $k \in G$, we have:

$$|\text{Prob}(D(1^k, z_k, a, \overrightarrow{XY}(1^k, a)) = 1) - \text{Prob}(D(1^k, z_k, a, \sigma(\overrightarrow{XY}(1^k, a)) = 1)| \geq 1/\mathsf{Poly}(k). \tag{1}$$

Consider $V \overset{def}{=} \{i : i \in [t], \sigma(i) \notin [t]\}$. We list the elements of V in order as $i_1 < \ldots < i_j \ldots < i_{|V|}$. Let $V_j \overset{def}{=} \{i_1, \ldots, i_j\}$. We define the following $|V| + 1$ permutations over $[n]$.

$$\phi_0(i) = \begin{cases} i & \text{if } i \in V \cup \{\sigma(i')\}_{i'\in V}, \\ \sigma(i) & \text{otherwise.} \end{cases}$$

For each $j \in [|V|]$, we set:

$$\phi_j(i) = \begin{cases} i & \text{if } i \in (V \setminus V_j) \cup \{\sigma(i')\}_{i'\in V\setminus V_j}, \\ \sigma(i) & \text{otherwise.} \end{cases}$$

In particular: $\sigma = \phi_{|V|}$. Note that $\phi_j(\overrightarrow{XY}(1^k, a))$ is well defined as $\sigma(\overrightarrow{XY}(1^k, a))$, because ϕ_j is a permutation over $[n]$ as σ. As $\phi_0(\overrightarrow{XY}(1^k, a))$ is obtained by not swapping positions between $X(1^k, a)$ and $Y(1^k, a)$, it holds that $\overrightarrow{XY}(1^k, a) \equiv \phi_0(\overrightarrow{XY}(1^k, a))$. So, we have:

$$\begin{aligned} &|\text{Prob}(D(1^k, z_k, a, \overrightarrow{XY}(1^k, a)) = 1) - \text{Prob}(D(1^k, z_k, a, \sigma(\overrightarrow{XY}(1^k, a))) = 1)| \\ &\qquad = |\text{Prob}(D(1^k, z_k, a, \phi_0(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_{|V|}(\overrightarrow{XY}(1^k, a))) = 1)|. \end{aligned} \tag{2}$$

Using the triangle inequality, we get:

$$\begin{aligned} &|\text{Prob}(D(1^k, z_k, a, \phi_0(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_{|V|}(\overrightarrow{XY}(1^k, a))) = 1)| \\ &\qquad \leq \sum_{j=1}^{|V|} |\text{Prob}(D(1^k, z_k, a, \phi_{j-1}(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_j(\overrightarrow{XY}(1^k, a))) = 1)|. \end{aligned} \tag{3}$$

Combining Inequality (1) to Inequality (3), we have:

$$\sum_{j=1}^{|V|} |\text{Prob}(D(1^k, z_k, a, \phi_{j-1}(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_j(\overrightarrow{XY}(1^k, a))) = 1)| \geq 1/\mathsf{Poly}(k).$$

So, there exists $j \in [|V|]$ such that

$$|\text{Prob}(D(1^k, z_k, a, \phi_{j-1}(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_j(\overrightarrow{XY}(1^k, a))) = 1)| \geq 1/(|V|\,\mathsf{Poly}(k)). \quad (4)$$

For any pair of permutations (ϕ_{j-1}, ϕ_j), they differ at values of points i_j and $\sigma(i_j)$. Similarly, for probability ensembles $\phi_{j-1}(\overrightarrow{XY}(1^k, a))$ and $\phi_j(\overrightarrow{XY}(1^k, a))$ differ at their i_j-th entry and $\sigma(i_j)$-th entry. Specifically,

$$\begin{aligned}
\phi_{j-1}(\overrightarrow{XY}(1^k, a))\langle i_j \rangle &\equiv X(1^k, a),\\
\phi_{j-1}(\overrightarrow{XY}(1^k, a))\langle \sigma(i_j) \rangle &\equiv Y(1^k, a),\\
\phi_{j}(\overrightarrow{XY}(1^k, a))\langle i_j \rangle &\equiv Y(1^k, a),\\
\phi_{j}(\overrightarrow{XY}(1^k, a))\langle \sigma(i_j) \rangle &\equiv X(1^k, a).
\end{aligned}$$

Let $\overrightarrow{MXY} \overset{def}{=} \{\overrightarrow{MXY}(1^k, a)\}_{k\in\mathbb{N}, a\in\{0,1\}^*}$ where $\overrightarrow{MXY}(1^k, a)$ is defined as follows.

$$\forall d \in [n] \quad \overrightarrow{MXY}(1^k, a)\langle d \rangle \equiv \begin{cases} \phi_{j-1}(\overrightarrow{XY}(1^k, a))\langle d \rangle & \text{if } d \neq \sigma(i_j) \\ X(1^k, a) & \text{if } d = \sigma(i_j) \end{cases}$$

Using the triangle inequality, we get:

$$\begin{aligned}
&|\text{Prob}(D(1^k, z_k, a, \phi_{j-1}(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \overrightarrow{MXY}(1^k, a)) = 1)| + \\
&\quad |\text{Prob}(D(1^k, z_k, a, \overrightarrow{MXY}(1^k, a)) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_j(\overrightarrow{XY}(1^k, a))) = 1)| \\
&\quad\quad \geq |\text{Prob}(D(1^k, z_k, a, \phi_{j-1}(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_j(\overrightarrow{XY}(1^k, a))) = 1)| \quad (5)
\end{aligned}$$

Combining Inequality (4) and Inequality(5), we know that either

$$|\text{Prob}(D(1^k, z_k, a, \phi_{j-1}(\overrightarrow{XY}(1^k, a))) = 1) - \text{Prob}(D(1^k, z_k, a, \overrightarrow{MXY}(1^k, a)) = 1)| \geq 1/(2\,|V|\,\mathsf{Poly}(k)) \quad (6)$$

or

$$|\text{Prob}(D(1^k, z_k, a, \overrightarrow{MXY}(1^k, a)) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_j(\overrightarrow{XY}(1^k, a))) = 1)| \geq 1/(2\,|V|\,\mathsf{Poly}(k)) \quad (7)$$

Without loss of generality, we assume that Inequality (6) holds (in the case where Inequality (7) holds, the proof is similar). The difference between $\overrightarrow{MXY}(1^k, a)$ and $\phi_{j-1}(\overrightarrow{XY}(1^k, a))$ is on their $\sigma(i_j)$-th entry. Specifically:

$$\begin{aligned}
\overrightarrow{MXY}(1^k, a)\langle \sigma(i_j) \rangle &\equiv X(1^k, a),\\
\phi_{j-1}(\overrightarrow{XY}(1^k, a))\langle \sigma(i_j) \rangle &\equiv Y(1^k, a).
\end{aligned}$$

We can construct a distinguisher D' with auxiliary input $z = (z_k)_{k\in\mathbb{N}}$ for the probability ensembles X and Y as follows.

- $D'(1^k, z_k, a, \gamma)$: Sample an instance vector $\overrightarrow{mxy}$ according to the random variable $\overrightarrow{MXY}(1^k, a)$ and set γ to be its $\sigma(i_j)$-th entry. Return $D'(1^k, z_k, a, \overrightarrow{mxy})$.

Obviously, we have:

$$\begin{aligned}
&|\text{Prob}(D'(1^k, z_k, a, X(1^k, a)) = 1) - \text{Prob}(D'(1^k, z_k, a, Y(1^k, a)) = 1)| = \\
&\quad |\text{Prob}(D(1^k, z_k, a, \overrightarrow{MXY}(1^k, a)) = 1) - \text{Prob}(D(1^k, z_k, a, \phi_{j-1}(\overrightarrow{XY}(1^k, a))) = 1)|. \quad (8)
\end{aligned}$$

Combining Inequality (6) and Inequality(8), we obtain:

$$|\text{Prob}(D'(1^k, z_k, a, X(1^k, a)) = 1) - \text{Prob}(D'(1^k, z_k, a, Y(1^k, a)) = 1)| \geq 1/(2\,|V|\,\mathsf{Poly}(k))$$

which contradicts the fact $X \overset{c}{=} Y$. $\square$

Remark 35. *Let* $\mathsf{Poly}_1(.), \mathsf{Poly}_2(.)$ *be two arbitrary positive polynomials such that* $\mathsf{Poly}_1(.) < \mathsf{Poly}_2(.)$*. It is easy to verify that Lemma 34 also holds in the case where* $t = \mathsf{Poly}_1(k)$*,* $n = \mathsf{Poly}_2(k)$*.*

We are ready to prove Theorem 28.

Proof. We reuse the notations HS_i from Definition 20, and BHS_i from Definition 26. For a hash family $\overline{\mathcal{H}}$ in Construction 1, we define the following probability ensemble for each $j \in [n]$.

$$\mathsf{HS}^j{=}\{\mathsf{HS}^j(1^k)\}_{k\in\mathbb{N}} \overset{def}{=} \{(\mathsf{HS}_1(1^k)\langle 1\rangle, \mathsf{HS}_1(1^k)\langle 2\rangle\langle j\rangle)\}_{k\in\mathbb{N}},$$

Recall that $\mathsf{HS}_1(1^k)\langle 1\rangle$ corresponds to the hash parameter Λ, and $\mathsf{HS}_1(1^k)\langle 2\rangle$ corresponds to the instance vector $\vec{x}^{\vec{a}}$. It is easy to see that: $\forall j \in [t]$ $\mathsf{HS}^j \equiv \mathsf{BHS}_1$ and: $\forall j \notin [t]$ $\mathsf{HS}^j \equiv \mathsf{BHS}_2$. Combining the fact that basic hash family $\mathcal{H}$ holds the hard subset membership property (i.e., $\mathsf{BHS}_1 \overset{c}{=} \mathsf{BHS}_2$) and Lemma 34, we have:

$$((\mathsf{HS}_1(1^k)\langle 1\rangle, \mathsf{HS}_1(1^k)\langle 2\rangle\langle 1\rangle), \ldots (\mathsf{HS}_1(1^k)\langle 1\rangle, \mathsf{HS}_1(1^k)\langle 2\rangle\langle n\rangle)) \overset{c}{=} \\ (\mathsf{HS}_2(1^k)\langle 1\rangle, \mathsf{HS}_2(1^k)\langle 2\rangle\langle 1\rangle), \ldots (\mathsf{HS}_2(1^k)\langle 1\rangle, \mathsf{HS}_2(1^k)\langle 2\rangle\langle n\rangle)).$$

Since $\mathsf{HS}_1(1^k)\langle 1\rangle \equiv \mathsf{HS}_2(1^k)\langle 1\rangle$, we have:

$$(\mathsf{HS}_1(1^k)\langle 1\rangle, \mathsf{HS}_1(1^k)\langle 2\rangle\langle 1\rangle, \ldots, \mathsf{HS}_1(1^k)\langle 2\rangle\langle n\rangle) \overset{c}{=} (\mathsf{HS}_2(1^k)\langle 1\rangle, \mathsf{HS}_2(1^k)\langle 2\rangle\langle 1\rangle, \ldots, \mathsf{HS}_2(1^k)\langle 2\rangle\langle n\rangle),$$

That is: $\mathsf{HS}_1 \overset{c}{=} \mathsf{HS}_2$. □

5. Constructing Basic Hash Families

In the previous section, we reduced the existence of a (n,t)-SPH-DHM to the construction of a basic hash family having smoothness (or universality), projection, distinguishability and hard subset membership. In this section, we show how to create such families under several mathematical hardness assumptions: DDH, LWE, DNR and DQR.

5.1. Instantiation under the Decisional Diffie-Hellman Assumption

Let $\mathsf{Gen}(1^k)$ be a PPT algorithm taking a security parameter k as input and outputting the description of a cyclic group G of prime order: $(g_1, g_2, q) \leftarrow \mathsf{Gen}(1^k)$ where g_1, g_2 and q are respectively two distinct generators of G and the group order.

Assumption 36. *The* decisional Diffie-Hellman (DDH) assumption *assumes that there is no non-uniform PPT algorithm distinguishing the following two probability ensembles* $\mathsf{DDH}_1{=}\{\mathsf{DDH}_1(1^k)\}_{k\in\mathbb{N}}$ *and* $\mathsf{DDH}_2{=}\{\mathsf{DDH}_2(1^k)\}_{k\in\mathbb{N}}$ *defined as:*

- $\mathsf{DDH}_1(1^k)$*: Set* $(g_1, g_2, q) \leftarrow \mathsf{Gen}(1^k)$ *and choose* $a \in_U \mathbb{Z}_q$*,* $b \in_U \mathbb{Z}_q$*. Output* $((g_1, g_2, q), g_1^a, g_2^b)$*.*
- $\mathsf{DDH}_2(1^k)$*: Operate as* $\mathsf{DDH}_1(1^k)$ *but return* $((g_1, g_2, q), g_1^a, g_2^a)$*.*

Our DDH-based basic hash family whose syntax is specified in Section 4.1 is described in Construction 2.

Lemma 37. *The hash family obtained from Construction 2 has perfect smoothness.*

Proof. Let $\Lambda = (g_1, g_2, q)$ be a legal hash parameter. Let $(\ddot{x}, a) = ((g_1^a, g_2^b), a)$ be an instance-witness pair generated by $\mathsf{IS}(1^k, \Lambda, 1)$. For this family, the probability ensembles BSM_1, BSM_2 in Definition 23 can be described as follows.

- $\mathsf{BSM}_1(1^k)$: Choose $u \in_U \mathbb{Z}_q$ and $v \in_U \mathbb{Z}_q$. Set $hk \leftarrow (u, v)$, $pk \leftarrow g_1^u g_2^v$ and $y \leftarrow g_1^{au} g_2^{bv}$. Output (pk, y).
- $\mathsf{BSM}_2(1^k)$: Run as $\mathsf{BSM}_1(1^k)$ with the exception that $y \in_U G$, where G is the group described by (g_1, g_2, q).

Construction 2 A DDH-based basic hash family

- $\mathsf{PG}(1^k)$: Set $\Lambda \leftarrow \mathsf{Gen}(1^k)$ and return Λ.
- $\mathsf{IS}(1^k, \Lambda, \delta)$: Parse Λ as (g_1, g_2, q) and choose $a \in_U \mathbb{Z}_q$, $b \in_U \mathbb{Z}_q$. Set $\dot{x} \leftarrow (g_1^a, g_2^a)$, $\dot{w} \leftarrow a$, $\ddot{x} \leftarrow (g_1^a, g_2^b)$ and $\ddot{w} \leftarrow a$. Return $(\dot{x}, \dot{w})$ if $\delta = 0$ or $(\ddot{x}, \ddot{w})$ if $\delta = 1$.
- $\mathsf{Check}(1^k, \Lambda)$: Parse Λ as (g_1, g_2, q). If q is a prime number of appropriate bitlength, and g_1, g_2 are two distinct generators, then outputs 1. Otherwise, output 0.
- $\mathsf{DI}(1^k, \Lambda, x, w)$: Parse Λ as (g_1, g_2, q) and x as (α,β). If $(\alpha,\beta) = (g_1^w, g_2^w)$, then output 0. If $\alpha = g_1^w$ and $\beta \neq g_2^w$, then output 1. Otherwise return 2.
- $\mathsf{KG}(1^k, \Lambda, x)$: Parse Λ as (g_1, g_2, q) and choose $u \in_U \mathbb{Z}_q$, $v \in_U \mathbb{Z}_q$. Set $hk \leftarrow (u, v)$ and compute $pk \leftarrow g_1^u g_2^v$. Output (hk, pk).
- $\mathsf{Hash}(1^k, \Lambda, x, hk)$: Parse x as (α,β) and hk as (u, v). Compute $y \leftarrow \alpha^u \beta^v$ and return y.
- $\mathsf{pHash}(1^k, \Lambda, x, pk, w)$: Compute $y \leftarrow pk^w$ and output y.

The difference between the two ensembles is the generation of y. Let's consider the value y generated by BSM_1. Since g_1 is a generator of the cyclic group G, there exists a constant c such that $g_2 = g_1^c$. Then: $g_1^{au} g_2^{bv} = g_1^{au+cbv}$. Since u and v are chosen uniformly from $\mathbb{Z}_q$, $au + cbv$ is also uniformly distributed over $\mathbb{Z}_q$. Since q is prime, $g_1^{au+cbv}(= y)$ is uniformly distributed over the group G. Therefore: $\mathsf{BSM}_1 \equiv \mathsf{BSM}_2$ since the value y created by $\mathsf{BSM}_2(1^k)$ is, by construction, uniformly uniformly distributed over G. □

Lemma 38. *The hash family obtained from Construction 2 has the projection property.*

Proof. Write $\Lambda = (g_1, g_2, q)$ and let $(\dot{x}, \dot{w}) = ((g_1^a, g_2^a), a)$ denote an instance-witness pair generated by $\mathsf{IS}(1^k, \Lambda, 0)$. Let $(hk, pk) = ((u, v), g_1^u g_2^v)$ be a hash-projection key pair generated by $\mathsf{KG}(1^k, \Lambda, \dot{x})$. We have:

$$\begin{aligned}\mathsf{Hash}(1^k, \Lambda, \dot{x}, hk) &= \mathsf{Hash}(1^k, \Lambda, (g_1^a, g_2^a), (u, v)) \\ &= g_1^{au} g_2^{av}, \\ \mathsf{pHash}(1^k, \Lambda, \dot{x}, pk, \dot{w}) &= \mathsf{pHash}(1^k, \Lambda, (g_1^a, g_2^a), g_1^u g_2^v, a) \\ &= g_1^{au} g_2^{av}.\end{aligned}$$

Therefore, we obtain: $\mathsf{Hash}(1^k, \Lambda, \dot{x}, hk) = \mathsf{pHash}(1^k, \Lambda, \dot{x}, pk, \dot{w})$. □

Lemma 39. *The hash family obtained from Construction 2 has the distinguishability property.*

The proof of this lemma is trivial, so we omit it.

Lemma 40. *Assuming that the DDH assumption holds, the hash family obtained from Construction 2 has hard subset membership.*

Proof. For this family, the probability ensembles BHS_1, BHS_2 in Definition 26 can be described as follows.

- $\mathsf{BHS}_1(1^k)$: Generate $\Lambda \leftarrow \mathsf{PG}(1^k)$, and parse Λ as (g_1, g_2, q). Choose $a \in_U \mathbb{Z}_q$ and compute $\dot{x} \leftarrow (g_1^a, g_2^a)$. Return $(\Lambda, \dot{x})$.
- $\mathsf{BHS}_2(1^k)$: Generate $\Lambda \leftarrow \mathsf{PG}(1^k)$, and parse Λ as (g_1, g_2, q). Choose $a \in_U \mathbb{Z}_q$, $b \in_U \mathbb{Z}_q$ and compute $\ddot{x} \leftarrow (g_1^a, g_2^b)$. Return $(\Lambda, \ddot{x})$.

Obviously: $\mathsf{BHS}_1 \overset{c}{=} \mathsf{BHS}_2$. □

5.2. Instantiation under the Learning with Errors Assumption

We start this section with some preliminaries related to the discretization of probabilistic distributions.

We consider the quotient ring $\mathbb{R}/\mathbb{Z}$ representing the segment $[0, 1)$ with addition modulo 1. Let β be an arbitrary positive real number. We denote by Ψ_β the distribution over $\mathbb{R}/\mathbb{Z}$ obtained by sampling from a normal variable with mean 0 and standard deviation $\beta/\sqrt{2\pi}$ and reducing the result modulo 1.

$$\begin{array}{rccl} \Psi_\beta : & \mathbb{R}/\mathbb{Z} & \longrightarrow & \mathbb{R}^+ \\ & r & \longmapsto & \displaystyle\sum_{k=-\infty}^{\infty} \frac{1}{\beta} \exp\left(-\pi\left(\frac{r-k}{\beta}\right)^2\right). \end{array}$$

Definition 41 ([38])**.** *Given an arbitrary integer $q \geq 2$, an arbitrary distribution $\phi : \mathbb{R}/\mathbb{Z} \to \mathbb{R}^+$, the* discretization *of ϕ over $\mathbb{Z}_q$ is defined as:*

$$\begin{array}{rccl} \overline{\phi} : & \mathbb{Z}_q & \longrightarrow & \mathbb{R}^+ \\ & i & \longmapsto & \displaystyle\int_{(i-1/2)/q}^{(i+1/2)/q} \phi(x)\, dx. \end{array}$$

Let $\mathsf{Gen}(1^k)$ be a PPT algorithm taking a security parameter k as input and outputs a description (q, m, χ), where q is a prime, $m = \mathsf{Poly}(k)$ and $\chi = \overline{\Psi}_\alpha$ where $\alpha \in (2\sqrt{k}/q, 1)$.

Assumption 42 ([38])**.** *The* decision version of the learning with errors (LWE) assumption *assumes that that there is no non-uniform PPT algorithm distinguishing the following two probability ensembles* LWE_1=$\{\mathsf{LWE}_1(1^k)\}_{k\in\mathbb{N}}$ *and* LWE_2=$\{\mathsf{LWE}_2(1^k)\}_{k\in\mathbb{N}}$ *defined as:*

- $\mathsf{LWE}_1(1^k)$*: Set $(q, m, \chi) \leftarrow$* $\mathsf{Gen}(1^k)$ *and choose $A \in_U (\mathbb{Z}_q)^{m\times k}$, $\vec{s} \in_U (\mathbb{Z}_q)^k$, $\vec{e} \in_\chi (\mathbb{Z}_q)^m$ (which means each entry of $\vec{e}$ is independently drawn from $\mathbb{Z}_q$ according to χ). Compute $\vec{b} \leftarrow A\vec{s} + \vec{e} \bmod q$ and output $((q, m, \chi), A, \vec{b})$.*
- $\mathsf{LWE}_2(1^k)$*: Operate as* $\mathsf{LWE}_1(1^k)$ *except that $\vec{b} \in_U (\mathbb{Z}_q)^m$.*

Remark 43. *Assumption 42 describes only the decision version of LWE problem. Nonetheless, the hardness of this decision problem is the same as the LWE problem itself [38].*

Remark 44. *The LWE problem is an average-case problem. However, Regev showed that its hardness is implied by both the worst-case hardness of the* shortest independent vector problem *(SIVP) and the worst-case hardness of the* decision version of the shortest vector problem *(GapSVP) (see Theorem 1 of [38]). In other words, if the LWE problem is efficiently solvable, so are both the SIVP and GapSVP. Since these two problems are believed to be hard even for quantum algorithms [39], so is the LWE problem.*

Before presenting our LWE-based instantiation of basic hash family, we need to recall the key generation algorithm of a LWE-based public key cryptosystem presented by Gentry *et al.* to be used as a building block [17].

- Message space: $\{0, 1\}$.
- $\mathsf{Setup}(1^k)$: Uniformly chooses a prime q such that $q \in [k^2, 2k^2]$. Set $m \leftarrow (1 + \varepsilon)(k + 1)\log q$ (where $\varepsilon > 0$ is an arbitrary constant and arbitrarily choose α such that $\alpha = o(1/(\sqrt{k}\log k))$ (e.g., $\alpha = \frac{1}{\sqrt{k}(\log k)^2}$). Set $\chi \leftarrow \overline{\Psi}_\alpha$ and $\Lambda \leftarrow (q, m, \chi)$. Output Λ.
- $\mathsf{KeyGen}(1^k, \Lambda)$: Parse Λ as (q, m, χ). Choose $A \in_U (\mathbb{Z}_q)^{m\times k}$, $\vec{s} \in_U (\mathbb{Z}_q)^k$ and $\vec{e} \in_\chi (\mathbb{Z}_q)^m$. Compute $\vec{b} \leftarrow A\vec{s} + \vec{e} \bmod q$ and set *pubk* $\leftarrow (A, \vec{b})$ and *sk* $\leftarrow \vec{s}$. Return the public-private key pair (*pubk*, *sk*).

Definition 45 ([17])**.** *A public key $(A, \vec{b})$ is* messy *if and only if, for all $m_0, m_1 \in \{0, 1\}$, the statistical difference between the distributions of* $\mathsf{Enc}_{A,\vec{b}}(m_0)$ *and* $\mathsf{Enc}_{A,\vec{b}}(m_1)$ *is negligible where* Enc *denotes the encryption algorithm of the cryptosystem.*

We would like to point out that any ciphertext produced under messy public keys carries no information (statistically) about the encrypted message. Therefore, for any (even unbounded) adversary, recovery of the corresponding plaintext is impossible. Gentry *et al.* showed that, if the construction parameters were appropriately chosen, their cryptosystem held the following properties.

P1 It provides security against chosen plaintext attack. [For our construction, we only need semantic security.]

P2 All but an at most $2/q^k$-fraction of $(A, \vec{b}) \in ((\mathbb{Z}_q)^{m\times k}, (\mathbb{Z}_q)^m)$ is messy.

P3 There exists an efficient algorithm $\mathsf{IsMessy}$ that identifies messy public keys. It takes a public key $(A, \vec{b})$, a trapdoor T as input and outputs an indicator. $\mathsf{IsMessy}$ has the following properties.

- $\mathsf{IsMessy}$ outputs 1 on an overwhelming fraction of all possible public keys which may be maliciously generated.
- If $\mathsf{IsMessy}(A, T, \vec{b})$ outputs 1 on a public key $(A, \vec{b})$, then $(A, \vec{b})$ is indeed messy.

We do not know how to gain distinguishability as defined in Definition 13. Instead, we will be able to obtained a relaxed version defined as follows.

Definition 46 (Relaxed Distinguishability)**.** *For any legal hash parameter* Λ*, any instance-witness pair* $(x, w) \in (\{0,1\}^*)^2$ *which may be maliciously generated, we require:*

$$\mathsf{DI}(1^k, \Lambda, x, w) = \begin{cases} 0 & \textit{if } (x, w) \in \dot{R}_\Lambda, \\ 1 & \textit{if } (x, w) \in \ddot{R}_\Lambda, \\ 0 \textit{ or } 2 & \textit{otherwise.} \end{cases}$$

In our framework for OT^n_t, the only usage of DI is to check the legalities of instance-witness vectors. According to the legality definition (Definition 17), an instance-witness vector is called legal if and only if it has at least $n - t$ entries that on receiving them DI outputs 1. Therefore, this relaxed distinguishability is sufficient for our objective.

Our LWE-based basic hash family whose syntax is specified in Section 4.1 is described in Construction 3.

Lemma 47. *The hash family obtained from Construction 3 has smoothness.*

Proof. Let $\Lambda = (q, m, \chi)$ be a legal hash parameter. Let $(\ddot{x}, \ddot{w}) = ((A, \vec{b}), (1, T))$ be an instance-witness pair generated by $\mathsf{IS}(1^k, \Lambda, 1)$. For this family, the probability ensembles BSM_1 and BSM_2 from Definition 23 can be described as follows.

- $\mathsf{BSM}_1(1^k)$: Choose $a \in_U \{0, 1\}$ and compute $pk \leftarrow \mathsf{Enc}_{A,\vec{b}}(a)$. Set $y \leftarrow a$ and output (pk, y).
- $\mathsf{BSM}_2(1^k)$: Choose $a' \in_U \{0, 1\}$ and compute $pk \leftarrow \mathsf{Enc}_{A,\vec{b}}(a')$. Pick $y \in_U \{0, 1\}$ and output (pk, y).

It is easy to see that the second component of each ensemble is uniformly distributed. The difference between these two ensembles is their first value denoted by pk. In the case of $\mathsf{BSM}_1(1^k)$, it is the encryption of y while, for $\mathsf{BSM}_2(1^k)$, it corresponds to the encryption of a'. Because $(A, \vec{b})$ is messy, the two distributions of this value pk are statistically indistinguishable. Therefore, $\mathsf{BSM}_1 \overset{s}{=} \mathsf{BSM}_2$. □

Lemma 48. *The hash family obtained from Construction 3 has the projection property.*

Proof. Let $(\dot{x}, \dot{w}) = ((A, \vec{b}), (0, \vec{e}, \vec{s}))$ be an instance-witness generated by $\mathsf{IS}(1^k, \Lambda, 0)$. Obviously, $((A, \vec{b}), \vec{s})$ is a normal public-private key pair. Then, we have:

$$\mathsf{Hash}(1^k, \Lambda, \dot{x}, hk) = a$$

and

$$\mathsf{pHash}(1^k, \Lambda, \dot{x}, pk, \dot{w}) = \mathsf{Dec}_{\vec{s}}(\alpha) = \mathsf{Dec}_{\vec{s}}(\mathsf{Enc}_{A,\vec{b}}(a)) = a$$

This means that, for any $(\Lambda, \dot{x}, \dot{w})$ generated by the hash family, it holds: $\mathsf{Hash}(1^k, \Lambda, \dot{x}, hk) = \mathsf{pHash}(1^k, \Lambda, \dot{x}, pk, \dot{w})$. □

Lemma 49. *The hash family obtained from Construction 3 has the distinguishability property.*

Construction 3 A LWE-based basic hash family

(Setup, KeyGen, Enc, Dec) representing the cryptosystem from [17] where the parameters of Setup were tuned so that properties P1, P2 and P3 hold.

- PG(1^k): Set $\Lambda \leftarrow \mathsf{Setup}(1^k)$ and return Λ.
- IS($1^k, \Lambda, \delta$): Parse Λ as (q, m, χ) and choose $A \in_U (\mathbb{Z}_q)^{m \times k}$ along with its trapdoor T, $\vec{s} \in_U (\mathbb{Z}_q)^k$ and $\vec{e} \in_\chi (\mathbb{Z}_q)^m$. Compute $\dot{x} \leftarrow (A, A\vec{s} + \vec{e} \bmod q)$ and set $\dot{w} \leftarrow (0, \vec{e}, \vec{s})$. Pick $\vec{b} \in_U (\mathbb{Z}_q)^m$ such that $\mathsf{IsMessy}(A, T, \vec{b}) = 1$ and set $\ddot{x} \leftarrow (A, \vec{b})$, $\ddot{w} \leftarrow (1, T)$. Output $(\dot{x}, \dot{w})$ if $\delta = 0$, $(\ddot{x}, \ddot{w})$ if $\delta = 1$.
- Check($1^k, \Lambda$): Parse Λ as (q, m, χ). If q is a prime number, m is a positive integer, and χ is a discretization of an appropriate distribution then output 1. Otherwise, return 0.
- DI($1^k, \Lambda, x, w$): Parse Λ as (q, m, χ) and x as $(A, \vec{b})$. Let i denote the value of w's first entry. If $i = 0$, then $(0, \vec{e}, \vec{s}) \leftarrow w$; if $i = 1$, then $(1, T) \leftarrow w$; otherwise output 2. Perform the following checks:
  - if $i = 0$: Check $\vec{b}, \vec{e} \in (\mathbb{Z}_q)^m$, $A \in (\mathbb{Z}_q)^{m \times k}$, $\vec{s} \in (\mathbb{Z}_q)^k$ and $\vec{b} = A\vec{s} + \vec{e} \bmod q$. If the checks are all successful then output 0; else output 2.
  - if $i = 1$: Check $\vec{b} \in (\mathbb{Z}_q)^m$, $A \in (\mathbb{Z}_q)^{m \times k}$, T is a trapdoor and $\mathsf{IsMessy}(A, T, \vec{b}) = 1$. If the checks are all successful then output 1; otherwise output 2.
- KG($1^k, \Lambda, x$): Parse Λ as (q, m, χ) and x as $(A, \vec{b})$. Choose $a \in_U \{0, 1\}$ and compute $\alpha \leftarrow \mathsf{Enc}_{A,\vec{b}}(a)$. Set $hk \leftarrow a$, $pk \leftarrow \alpha$ and output (hk, pk).
- Hash($1^k, \Lambda, x, hk$): Output hk.
- pHash($1^k, \Lambda, x, pk, w$): Parse Λ as (m, q, χ) and w as $(0, \vec{e}, \vec{s})$. Compute $a \leftarrow \mathsf{Dec}_{\vec{s}}(pk)$ and output a.

Proof. Let Λ be a legal hash parameter, (x, w) an instance-witness pair that may be maliciously generated. It is trivial to see that DI outputs 0 if (x, w) can be generated by $\mathsf{IS}(1^k, \Lambda, 0)$ (i.e., $(x, w) \in \dot{R}_\Lambda$).

Recalling the property of algorithm $\mathsf{IsMessy}$, we know that if DI outputs 1, then $(x, w) \in \ddot{R}_\Lambda$; if $(x, w) \in \ddot{R}_\Lambda$, then DI outputs 1 with probability almost 1. □

Lemma 50. *Assuming the LWE assumption holds, the hash family obtained from Construction 3 has hard subset membership.*

Proof. For this family, the probability ensembles BHS_1, BHS_2 from Definition 26 can be described as follows.

- $\mathsf{BHS}_1(1^k)$: Generate $\Lambda \leftarrow \mathsf{PG}(1^k)$ and parse it as (q, m, χ). Choose $A \in_U (\mathbb{Z}_q)^{m\times k}$ along with its trapdoor T, $\vec{s} \in_U (\mathbb{Z}_q)^k$ and $\vec{e} \in_\chi (\mathbb{Z}_q)^m$. Compute $\dot{x} \leftarrow (A, A\vec{s} + \vec{e} \bmod q)$ and output $(\Lambda, \dot{x})$.
- $\mathsf{BHS}_2(1^k)$: Generate $\Lambda \leftarrow \mathsf{PG}(1^k)$ and parse it as (q, m, χ). Choose $A \in_U (\mathbb{Z}_q)^{m\times k}$ along with its trapdoor T and pick $\vec{b} \in_U (\mathbb{Z}_q)^m$ such that $\mathsf{IsMessy}(A, T, \vec{b}) = 1$. Set $\ddot{x} \leftarrow (A, \vec{b})$ and output $(\Lambda, \ddot{x})$.

Obviously, $\mathsf{BHS}_1 \overset{c}{=} \mathsf{BHS}_2$. □

Since, in our basic hash family, each instance holds a matrix A, so does the hash family resulting from Construction 1. It is trivial to show that this hash family can be improved by each instance vector sharing a common matrix A. However, to securely instantiate our framework for OT^n_t, it cannot be improved by all instance vectors sharing a common matrix A. The reason is that, in this case, seeing A's trapdoor T in Step S2 of the framework Π, the sender can distinguish smooth instances and projective instances of the unchosen instance vectors and thus he can deduce the receiver's private input.

5.3. Instantiation under the Decisional Quadratic Residuosity Assumption

Let $\mathsf{Gen}(1^k)$ be a PPT algorithm taking a security parameter k as input and returning an integer N being the product of two distinct k-bit odd primes. Let J_N be the subgroup of $\mathbb{Z}^*_N$ of elements having Jacobi symbol $+1$.

Assumption 51 ([40])**.** *The* decisional quadratic residuosity (DQR) assumption *assumes that there is no non-uniform PPT algorithm distinguishing the following two probability ensembles* $\mathsf{DQR}_1 = \{\mathsf{DQR}_1(1^k)\}_{k\in\mathbb{N}}$ *and* $\mathsf{DQR}_2 = \{\mathsf{DQR}_2(1^k)\}_{k\in\mathbb{N}}$ *defined as:*

- $\mathsf{DQR}_1(1^k)$*: Generate $N \leftarrow \mathsf{Gen}(1^k)$, choose $r \in_U \mathbb{Z}^*_N$ and compute $x \leftarrow r^2 \bmod N$. Output (N, x).*
- $\mathsf{DQR}_2(1^k)$*: Operate as $\mathsf{DQR}_1(1^k)$ except $x \in_U J_N$.*

Definition 52 ([16])**.** *A hash family $\mathcal{H} = (\mathsf{PG}, \widehat{\mathsf{IS}}, \widehat{\mathsf{IT}}, \widehat{\mathsf{KG}}, \mathsf{Hash}, \mathsf{pHash})$ is a* verifiably-ε-universal projective hashing family*, if for any Λ generated by PG(1^k), any $(\dot{x}, \dot{w}, \ddot{x})$ generated by $\widehat{\mathsf{IS}}(1^k, \Lambda)$, $\mathcal{H}$ is projective on $\dot{x}$, and universal on $\ddot{x}$. In addition, it holds verifiable-ε-universallity, which is defined as follows.*

- *For any Λ generated by $\mathsf{PG}(1^k)$, any $(\dot{w}, \dot{x}, \ddot{x})$ generated by $\widehat{\mathsf{IS}}(1^k, \Lambda)$, we have: $\widehat{\mathsf{IT}}(1^k, \Lambda, \dot{x}, \ddot{x}) = \widehat{\mathsf{IT}}(1^k, \Lambda, \ddot{x}, \dot{x}) = 1$.*
- *For any tuple (Λ, x_1, x_2) which may be maliciously generated, if $\widehat{\mathsf{IT}}(1^k, \Lambda, x_1, x_2) = 1$, then $\mathcal{H}$ is ε-universal on at least one of x_1 or x_2.*

We can see that the syntax of a hash family appearing in [16] is different from our basic hash family notion. It does not have algorithms like Check or DI. However, it possesses an algorithm $\widehat{\mathsf{IT}}$. There are some additional syntaxic differences not reflected by above definition. In a Halevi and Tauman Kalai hash family, $\widehat{\mathsf{IS}}$ does not take any work mode parameter δ and $\widehat{\mathsf{IS}}(1^k, \Lambda)$ outputs a tuple $(\dot{x}, \dot{w}, \ddot{x})$ where typically $\dot{x}$ and $\ddot{x}$ are generated in dependent way. In addition, their key generating algorithm $\widehat{\mathsf{KG}}$ does not take any instance parameter x.

Under the DQR assumption, Halevi and Tauman Kalai proposed the following verifiably-ε-universal projective hash family.

Construction 4 A verifiably-universal projective hash family [16]

- $\mathsf{PG}(1^k)$: Choose uniformly two k-bit prime p, q such that $p = q = 3 \bmod 4$. Set $N \leftarrow pq$, choose $a \in_U Z_N^*$ and compute $g \leftarrow a^2 \bmod N$. Set $\Lambda \leftarrow (N, g)$ and output Λ.
- $\widehat{\mathsf{IS}}(1^k, \Lambda)$: Parse Λ as (N, g). Set $T \leftarrow 2^{\lceil \log N \rceil}$, choose $\dot{w} \in_U \mathbb{Z}_N$ and compute $\dot{x} \leftarrow (g^T)^{\dot{w}} \bmod N$ and $\ddot{x} \leftarrow N - \dot{x}$. Output $(\dot{w}, \dot{x}, \ddot{x})$.
- $\widehat{\mathsf{IT}}(1^k, \Lambda, \dot{x}, \ddot{x})$: Parse Λ as (N, g). Checks that $N > 2^{2n}$, $g, \dot{x} \in Z_N^*$, and $\ddot{x} = N - \dot{x}$. Outputs 1 if all these tests pass and 0 otherwise.
- $\widehat{\mathsf{KG}}(1^k, \Lambda)$: Parse Λ as (N, g). Set $T \leftarrow 2^{\lceil \log N \rceil}$, choose $hk \in_U \mathbb{Z}_N$ and compute $pk \leftarrow (g^T)^{hk} \bmod N$. Output (hk, pk).
- $\mathsf{Hash}(1^k, \Lambda, x, hk)$: Parse Λ as (N, g) and compute $y \leftarrow x^{hk} \bmod N$. Output y.
- $\mathsf{pHash}(1^k, \Lambda, x, pk, w)$: Parse Λ as (N, g) and compute $y \leftarrow pk^w \bmod N$. Output y.

Our DQR-based basic hash family whose syntax is specified in Section4.1 is described in Construction 5.

Construction 5 A DQR-based basic hash family

- Algorithms PG, KG, Hash, pHash operate as in Construction 4.
- $\mathsf{IS}(1^k, \Lambda, \delta)$: Parse Λ as (N, g). Set $T \leftarrow 2^{\lceil \log N \rceil}$, choose $r \in_U \mathbb{Z}_N$ and compute $\dot{x} \leftarrow (g^T)^r \bmod N$. Set $\dot{w} \leftarrow r$, compute $\ddot{x} \leftarrow N - (g^T)^r \bmod N$ and assign $\ddot{w} \leftarrow r$. Output $(\dot{x}, \dot{w})$ if $\delta = 0$; output $(\ddot{x}, \ddot{w})$ if $\delta = 1$.
- $\mathsf{Check}(1^k, \Lambda)$: Parse Λ as (N, g). If it holds that $N > 2^{2k}$ and $g \in Z_N^*$, then output 1; otherwise, output 0.
- $\mathsf{DI}(1^k, \Lambda, x, w)$: Parse Λ as (N, g). Set $T \leftarrow 2^{\lceil \log N \rceil}$ and $r \leftarrow w$. If $x = (g^T)^r \bmod N$, then output 0. If $x = N - (g^T)^r \bmod N$, then output 1. Otherwise, output 2.

It is easy to see that the basic hash family obtained from Construction 5 inherits the universality and projection properties from Construction 4.

Lemma 53. *The hash family obtained from Construction 5 has the distinguishability property.*

Proof. Let Λ be a legal hash parameter and (x, w) an instance-witness pair that may be maliciously generated. It is easy seen that $\mathsf{DI}(1^k, \Lambda, x, w) = 0$ if and only if (x, w) can be generated by $\mathsf{IS}(1^k, \Lambda, 0)$ (i.e., $(x, w) \in \dot{R}_\Lambda$) and $\mathsf{DI}(1^k, \Lambda, x, w) = 1$, if and only if (x, w) can be generated by $\mathsf{IS}(1^k, \Lambda, 1)$ (i.e., $(x, w) \in \ddot{R}_\Lambda$). $\square$

Lemma 54. *Assuming that the DQR assumption holds, the hash family obtained from Construction 5 has hard subset membership.*

Proof. For this family, the probability ensembles BHS_1 and BHS_2 from Definition 26 can be described as follows.

- $\mathsf{BHS}_1(1^k)$: Generate $(N, g) \leftarrow \mathsf{PG}(1^k)$. Set $T \leftarrow 2^{\lceil \log N \rceil}$, choose $r \in_U \mathbb{Z}_N$ and compute $\dot{x} \leftarrow (g^T)^r \bmod N$. Output $(\Lambda, \dot{x})$.
- $\mathsf{BHS}_2(1^k)$: Operate as $\mathsf{BHS}_1(1^k)$ except that it outputs $(\Lambda, \ddot{x})$, where $\ddot{x} \leftarrow N - (g^T)^r \bmod N$.

It is easy to see that $(g^T)^r \bmod N$ is a quadratic residue modulo N and N is a Blum integer. Following the properties of Blum integers, $N - (g^T)^r \bmod N$ is not a quadratic residue and $N - (g^T)^r \bmod N \in J_N$. Therefore, if the DQR assumption holds, we have: $\mathsf{BHS}_1 \overset{c}{=} \mathsf{BHS}_2$. $\square$

5.4. Instantiation under the Decisional N-th Residuosity Assumption

We use the same parameter generator $\mathsf{Gen}(1^k)$ as in Section 5.3.

Assumption 55 ([41])**.** *The* decisional N-th residuosity (DNR) assumption *assumes that there is no non-uniform PPT algorithm distinguishing the following two probability ensembles* $\mathsf{DNR}_1 = \{\mathsf{DNR}_1(1^k)\}_{k\in\mathbb{N}}$ *and* $\mathsf{DNR}_2 = \{\mathsf{DNR}_2(1^k)\}_{k\in\mathbb{N}}$ *defined as follows:*

- $\mathsf{DNR}_1(1^k)$*: Generate* $N \leftarrow \mathsf{Gen}(1^k)$ *and choose* $a \in_U \mathbb{Z}^*_{N^2}$*. Compute* $b \leftarrow a^N \bmod N^2$ *and output* (N, b)*.*
- $\mathsf{DNR}_2(1^k)$*: Operate as* $\mathsf{DNR}_1(1^k)$ *except that* $b \in_U \mathbb{Z}^*_{N^2}$*.*

Similarly to our DQR-based basic hash family, our DNR-based basic hash family is built from Halevi and Kalai's DNR-based verifiably-ε-universal projective hash family [16]. The modifications and security proofs are similar too. To save space, we omit these details and only describe our basic hash family in Construction 6.

Construction 6 A DNR-based basic hash family

- $\mathsf{PG}(1^k)$: Generate $N \leftarrow \mathsf{Gen}(1^k)$, choose $a \in_U Z^*_{N^2}$ and compute $g \leftarrow a^N \bmod N$. Set $\Lambda \leftarrow (N, g)$ and return Λ.
- $\mathsf{IS}(1^k, \Lambda, \delta)$: Parse Λ as (N, g). Set $T \leftarrow N^{\lceil 2\log N\rceil}$, choose $r \in_U \mathbb{Z}^*_N$ and compute $\dot{x} \leftarrow (g^T)^r \bmod N^2$. Set $\dot{w} \leftarrow (r, 0)$, $v \in_U \mathbb{Z}^*_N$, choose $\ddot{x} \leftarrow (g^T)^r(1 + vN) \bmod N^2$ and $\ddot{w} \leftarrow (r, v)$. Output $(\dot{x}, \dot{w})$ if $\delta = 0$; output $(\ddot{x}, \ddot{w})$ if $\delta = 1$.
- $\mathsf{Check}(1^k, \Lambda)$: Parse Λ as (N, g). If $N > 2^{2k}$ and $g \in \mathbb{Z}^*_{N^2}$, then output 1. Otherwise, output 0.
- $\mathsf{DI}(1^k, \Lambda, x, w)$: Parse Λ as (N, g) and w as (r, v). If $v = 0$, $r \in \mathbb{Z}^*_N$ and $x = (g^T)^r \bmod N^2$, then output 0. If $v \neq 0$, $r, v \in \mathbb{Z}^*_N$ and $x = (g^T)^r(1 + vN) \bmod N^2$, then output 1. Otherwise, output 2.
- $\mathsf{KG}(1^k, \Lambda, x)$: Parse Λ as (N, g). Set $T \leftarrow N^{\lceil 2\log N\rceil}$, choose $hk \in_U \mathbb{Z}_{N^2}$ and compute $pk \leftarrow (g^T)^{hk} \bmod N^2$. Output (hk, pk).
- $\mathsf{Hash}(1^k, \Lambda, x, hk)$: Output $x^{hk} \bmod N^2$.
- $\mathsf{pHash}(1^k, \Lambda, x, pk, w)$: Output $pk^w \bmod N^2$.

Bibliography

References

[1] M. O. Rabin, How to exchange secrets by oblivious transfer, Tech. Rep. Technical Report TR-81, Aiken Computation Lab, Harvard University (1981).

[2] J. Kilian, Founding cryptography on oblivious transfer, in: 20th Annual ACM Symposium on Theory of Computing (STOC'88), ACM Press, Chicago, USA, 1988, pp. 20 – 31.

[3] Y. Lindell, B. Pinkas, Privacy preserving data mining, Journal of Cryptology 15 (2002) 177–206.

[4] M. Naor, B. Pinkas, Computationally secure oblivious transfer, Journal of Cryptology 18 (1) (2005) 1–35.

[5] M. Green, S. Hohenberger, Practical adaptive oblivious transfer from simple assumptions, in: Y. Ishai (Ed.), 8th Theory of Cryptography Conference (TCC'11), Vol. 6597, Springer - Verlag, Providence, USA, 2011, pp. 347 – 363.

[6] W. Ogata, K. Kurosawa, Oblivious keyword search, Journal of complexity 20 (2) (2004) 356–371.

[7] B. Aiello, Y. Ishai, O. Reingold, Priced oblivious transfer: How to sell digital goods, in: B. Pfitzmann (Ed.), Advances in Cryptology - Eurocrypt'01, Vol. 2045 of Lecture Notes in Computer Science, Springer - Verlag, Innsbruck, Austria, 2001, pp. 119 – 135.

[8] R. Cramer, V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in: L. R. Knudsen (Ed.), Advances in Cryptology - Eurocrypt'02, Vol. 2332 of Lecture Notes in Computer Science, Springer - Verlag, Amsterdam, The Netherlands, 2002, pp. 45 – 64.

[9] M. Naor, B. Pinkas, Efficient oblivious transfer protocols, in: 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01), Society for Industrial and Applied Mathematics, Washington, USA, 2001, pp. 448 – 457.

[10] M. Green, S. Hohenberger, Blind identity-based encryption and simulatable oblivious transfer, in: K. Kurosawa (Ed.), Advances in Cryptology - Asiacrypt'07, Vol. 4833 of Lecture Notes in Computer Science, Springer - Verlag, Kuching, Malaysia, 2007, pp. 265 – 282.

[11] J. Camenisch, G. Neven, abhi shelat, Simulatable adaptive oblivious transfer, in: M. Naor (Ed.), Advances in Cryptology - Eurocrypt'07, Vol. 4515 of Lecture Notes in Computer Science, Springer - Verlag, Barcelona, Spain, 2007, pp. 573 – 590.

[12] A. Y. Lindell, Efficient fully-simulatable oblivious transfer, in: T. Malkin (Ed.), Topics in Cryptology - CT-RSA 2008, Vol. 4964 of Lecture Notes in Computer Science, Springer - Verlag, San Francisco, USA, 2008, pp. 52 – 70.

[13] Y. Lindell, B. Pinkas, Secure two-party computation via cut-and-choose oblivious transfer, Journal of Cryptology 25 (4) (2012) 680–722.

[14] C. Peikert, V. Vaikuntanathan, B. Waters, A framework for efficient and composable oblivious transfer, in: D. Wagner (Ed.), Advances in Cryptology - Crypto'08, Vol. 5157 of Lecture Notes in Computer Science, Springer - Verlag, Santa Barbara, USA, 2008, pp. 554 – 571.

[15] B. Zeng, Founding cryptography on smooth projective hashing, Cryptology ePrint Archive, Report 2018/444, `https://eprint.iacr.org/2018/444` (2018).

[16] S. Halevi, Y. Tauman Kalai, Smooth projective hashing and two-message oblivious transfer, Journal of Cryptology 25 (1) (2012) 158–193.

[17] C. Gentry, C. Peikert, V. Vaikuntanathan, Trapdoors for hard lattices and new cryptographic constructions, in: C. Dwork (Ed.), 40th Annual ACM Symposium on Theory of Computing (STOC'08), ACM Press, Victoria, Canada, 2008, pp. 197 – 206.

[18] B. Zeng, C. Tartary, P. Xu, J. Jing, X. Tang, A practical framework for t-out-of-n oblivious transfer with security against covert adversaries, IEEE Transactions on Information Forensics and Security 7 (2) (2012) 465–479.

[19] S. G. Choi, J. Katz, H. Wee, H.-S. Zhou, Efficient, adaptively secure, and composable oblivious transfer with a single, global crs, in: K. Kurosawa, G. Hanaoka (Eds.), Public-Key Cryptography – PKC 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 73–88.

[20] M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, Sphf-friendly non-interactive commitments, in: K. Sako, P. Sarkar (Eds.), Advances in Cryptology - ASIACRYPT 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 214–234.

[21] O. Blazy, C. Chevalier, Structure-preserving smooth projective hashing, in: J. H. Cheon, T. Takagi (Eds.), Advances in Cryptology – ASIACRYPT 2016, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 339–369.

[22] M. Naor, G. Segev, Public-key cryptosystems resilient to key leakage, SIAM Journal on Computing 41 (4) (2012) 772–814.

[23] K. Kurosawa, R. Nojima, et al., New leakage-resilient cca-secure public key encryption, Journal of Mathematical Cryptology 7 (4) (2013) 297–312.

[24] B. Qin, S. Liu, Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter, in: K. Sako, P. Sarkar (Eds.), Advances in Cryptology - ASIACRYPT 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 381–400.

[25] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, D. Wichs, Public-key encryption in the bounded-retrieval model, in: H. Gilbert (Ed.), Advances in Cryptology – EUROCRYPT 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 113–134.

[26] H. Wee, Kdm-security via homomorphic smooth projective hashing, in: C.-M. Cheng, K.-M. Chung, G. Persiano, B.-Y. Yang (Eds.), Public-Key Cryptography – PKC 2016, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 159–179.

[27] S. M. Bellovin, M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, in: Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on, IEEE, 1992, pp. 72–84.

[28] R. Gennaro, Y. Lindell, A framework for password-based authenticated key exchange, ACM Transactions on Information and System Security (TISSEC) 9 (2) (2006) 234.

[29] J. Katz, R. Ostrovsky, M. Yung, Efficient password-authenticated key exchange using human-memorable passwords, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2001, pp. 475–494.

[30] R. Canetti, S. Halevi, J. Katz, Y. Lindell, P. MacKenzie, Universally composable password-based key exchange, in: R. Cramer (Ed.), Advances in Cryptology – EUROCRYPT 2005, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 404–421.

[31] M. Abdalla, C. Chevalier, D. Pointcheval, Smooth projective hashing for conditionally extractable commitments, in: S. Halevi (Ed.), Advances in Cryptology - CRYPTO 2009, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 671–689.

[32] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, D. Vergnaud, New techniques for sphfs and efficient one-round pake protocols, in: R. Canetti, J. A. Garay (Eds.), Advances in Cryptology – CRYPTO 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 449–475.

[33] J. Katz, V. Vaikuntanathan, Round-optimal password-based authenticated key exchange, in: Y. Ishai (Ed.), Theory of Cryptography, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 293–310.

[34] O. Goldreich, Foundations of Cryptography: Volume I - Basic Tools, Cambridge University Press, 2001.

[35] O. Goldreich, Foundations of Cryptography: Volume II - Basic Applications, Cambridge University Press, 2004.

[36] R. Canetti, Security and composition of multiparty cryptographic protocols, Journal of Cryptology 13 (1) (2000) 143 – 202.

[37] M. Luby, Pseudorandomness and Cryptographic Applications, Princeton University Press, 1996.

[38] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, Journal of the ACM (JACM) 56 (6).

[39] D. J. Bernstein, J. Buchmann, E. Dahmen (Eds.), Post-Quantum Cryptography, Springer, 2009.

[40] S. Goldwasser, S. Micali, Probabilistic encrytion, Journal of Computer and System Sciences 28 (2) (1984) 270 – 299.

[41] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: J. Stern (Ed.), Advances in Cryptology - Eurocrypt'99, Vol. 1592 of Lecture Notes in Computer Science, Springer - Verlag, Prague, Czech Republic, 1999, pp. 223 – 238.