Post-Quantum UC-Secure Oblivious Transfer in the Standard Model with Adaptive Corruptions

Olivier Blazy[⋆], Céline Chevalier[⋆⋆], and Quoc Huy Vu[⋆ ⋆ ⋆]

Abstract. Since the seminal result of Kilian, Oblivious Transfer has proven to be a fundamental primitive in cryptography. In such a scheme, a user is able to gain access to an element owned by a server, without learning more than this single element, and without the server learning which element the user has accessed. This primitive has received a lot of study in the literature, among which very few schemes are based on lattices. The recent NIST call for post-quantum encryption and signature schemes has revived the interest for cryptographic protocols based on post-quantum assumptions and the need for a secure post-quantum oblivious transfer scheme. In this paper, we show how to construct an oblivious transfer scheme based on lattices, from a collision-resistant chameleon hash scheme (CH) and a CCA encryption scheme accepting a smooth projective hash function (SPHF). Note that our scheme does not rely on random oracles and provides UC security against adaptive corruptions assuming reliable erasures.

Keywords. Post-Quantum Cryptography, Lattices, Commitments, Smooth Projective Hash Functions, CCA encryption, Oblivious Transfer, UC Framework.

1 Introduction

With the development of cloud services, sharing data securely and efficiently has never been so important. *Oblivious Transfer* was introduced in 1981 by Rabin [Rab81] and has proven to be a fundamental primitive in cryptography since the results on the completeness of oblivious transfer for multi-party computation, starting with the seminal work of Kilian [Kil88]. It is for example needed for every bit of input in Yao's protocol [Yao86] as well as for Oblivious RAM ([WHC$^+$14] for instance), for every AND gate in the Boolean circuit computing the function in [GMW87] or for almost all known garbled circuits [BHR12]. It has been widely used and studied in the community.

In its classical 1-out-of-N version, an OT protocol allows a user to ask access to a single line of a database (containing N lines of data) owned by a server. This exchange is oblivious in two ways: First, the privacy of the receiver is preserved in the sense that the sender learns no information on which element of the database has been asked for. Then, the privacy of the database is ensured in the sense that the user only gains access to the element he asked for and no information is linked on the others.

For practical purposes, there is undoubtedly a need for post-quantum instantiations of cryptographic protocols. While the recent call of the NIST mainly focuses on post-quantum encryption and signatures, we believe it is important to extend post-quantum

[⋆] Université de Limoges, XLIM, Limoges, France. olivier.blazy@unilim.fr

[⋆⋆] CRED, Université Panthéon-Assas, Paris, France. celine.chevalier@ens.fr

[⋆ ⋆ ⋆] DIENS, École normale supérieure, CNRS, INRIA, PSL University, Paris, France. quoc.huy.vu@ens.fr

research to two-party protocols and in particular oblivious transfer, which is an essential primitive that can be combined to fulfill important cryptographic tasks.

Building Blocks. In order to compose such bricks, the natural security model which arises is the *universal composability framework* proposed in [Can01]. In a nutshell, in the UC framework, security for a specific kind of protocol is captured by an ideal functionality (in an ideal world). A protocol is then proven secure if, given any adversary to the protocol in the real world, one can construct a simulator of this adversary in the ideal world, such that no environment can distinguish between the execution in the ideal world and the execution in the real world in a non-negligible way.

An additional difficulty arises when one wants to ensure *adaptive security*, which means that the adversary can corrupt the players at any moment during the execution of a protocol. In such situations, the usual trick is to use commitments with very strong properties. Commitment schemes are two-party primitives (between a committer and a receiver) divided into two phases. In the first *commit* phase, the committer gives the receiver an analogue of a sealed envelope containing a value m, while in the second *opening* phase, the committer reveals m in such a way that the receiver can verify that it was indeed m which was contained in the envelope. It is required that the committer cannot change the committed value (*binding* property) and that the receiver cannot learn anything about m before the opening phase (*hiding* property). It is impossible to perfectly achieve both properties (rather than computationally or statistically) at the same time. ElGamal [ElG84] or Cramer-Shoup [CS02] encryptions are famous examples of perfectly binding commitments, and Pedersen encryption [Ped92] is the most well-known example of perfectly hiding commitments. When speaking about UC adaptive security, the two additional properties are *extractability* (meaning that a simulator can recover the value committed to thanks to a trapdoor) and *equivocability* (meaning that a simulator can open a commitment to a value m' different from the value m it committed to thanks to another trapdoor).

Furthermore, in cases such as oblivious transfer in which the secrecy of the message (the number of the line required by the user) forbids the opening of the commitment, one has to use *implicit decommitment*. The now classical way to achieve this goal [CF01,ACP09,ABB$^+$13] is to combine these commitments with a smooth projective hash function (SPHF). They have been initially defined by Cramer and Shoup [CS02] and their first applications (to PAKE schemes) date back to [GL03,CHK$^+$05]. The outputs of these hash functions can be computed in two different ways if the input belongs to a particular subset (called the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language. The hash value obtained is indistinguishable from random in case the input does not belong to the language (*smoothness*) and in case the input does belong to the language but no witness is known (*pseudo-randomness*).

Related Work. Since the original paper [Rab81], several instantiations of OT protocols have appeared in the literature [NP01,CLOS02], including proposals in the UC framework. Some instantiations try to reach round-optimality [HK07], and/or low communication costs [PVW08]. Choi *et al.* [CKWZ13] proposed a generic method and an efficient instantiation secure against adaptive corruptions in the CRS model with erasures, but it does not scale to 1-out-of-N OT, for $N > 2$. Recent schemes like [ABB$^+$13,BC15,BC16,BCG17] manage to achieve round-optimality while maintaining a small communication cost.

However, among these articles, only the schemes from [PVW08] (an ad-hoc construction based on lattices) and [BC15] (a generic construction relying on [KV09] for the instantiation based on lattices) offer post-quantum security. Unfortunately, the first construction only fulfills static security. The second one offers adaptive security, but relies on the only standard-model lattice-based SPHF construction [KV09], which has the main drawback of defining the language of the SPHF as the set of all the ciphertexts such that at least one integer multiple is close to the public lattice rather than the set of valid standard LWE ciphertexts. A consequence is that the decryption procedure is very costly (about q trapdoor inversions) and forbids the use of superpolynomial modulus q. This is obviously not the same with classical SPHFs in a group-based setting, which can handle classical ElGamal or Cramer-Shoup encryption schemes, without any modification of the decryption procedure.

Our contributions. We describe in this paper an oblivious transfer scheme which is post-quantum (based on LWE), UC-secure, and deals with adaptive corruptions assuming reliable erasures. To the best of our knowledge, this is the first post-quantum OT scheme with such a high level of security. Our methodology relies on the generic construction of [BC15]. In order to instantiate the necessary building blocks, we replace the use of the SPHF construction of [KV09] by a chameleon hash function and an IND-CCA2 and an SPHF construction from [BBDQ18]. This allows us to give an SPHF-friendly commitment scheme based on LWE, which can be seen as a side contribution of the paper. Furthermore, we propose concrete parameters and an implementation of our scheme, which is, to the best of our knowledge, the first proof of concept of a post-quantum oblivious transfer scheme.

Roadmap of the Paper. In a preliminary section (Section 2 and Appendix A for details), we give all necessary definitions and properties of the primitives we use. Then we recall the lattice-based constructions needed for our instantiations in Section 3, which lead to our constructions of post-quantum SPHF-friendly commitment and OT scheme, presented in Sections 4 and 5 (the generic construction being recalled in Appendix B for completeness). Finally, concrete parameters are presented in Appendix C and a prototype will be made publicly available.

2 Notations and Definitions

Notations. Let $\kappa \in \mathbb{N}$ be the security parameter. We say that a function is *negligible* in κ, and we denote it by $\mathsf{negl}(\kappa)$, if it is a $f(\kappa) = \kappa^{-\omega(1)}$. When sampling uniformly at random the value a from the set U, we employ the notation $a \overset{\$}{\leftarrow} U$. When sampling the value a from the probabilistic algorithm χ, we employ the notation $a \leftarrow \chi$. For $a, b \in \mathbb{R}$, $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$ and $[\![a, b]\!] = \{x \in \mathbb{Z} \mid a \leq x \leq b\}$ will denote the closed real and integer interval with endpoints a and b, $\lfloor a \rfloor$, $\lceil a \rceil$, $\lfloor a \rceil$ will respectively denote the floor, ceiling and rounding function. The cardinal of a finite set S is denoted $|S|$.

Throughout this paper, we will work in a lattice-based setting. Here we describe the notations that will be used (see below for more definitions and details on lattices). We denote by n and m the rank and the dimension of a lattice, respectively. For simplicity, throughout this paper we use the base-2 logarithm, denoted by $\log$. Let $\omega(\sqrt{\log n})$ represent a fixed function that asymptotically grows faster than $\sqrt{\log n}$ [1].

Column vectors will be denoted by bold lower-case letters, *e.g.*, $\mathbf{x}$, and matrices will be denoted by bold upper-case letters, *e.g.*, $\mathbf{A}$. If $\mathbf{x}$ is a vector and $\mathbf{A}$ is a matrix, $\mathbf{x}^t$ and $\mathbf{A}^t$ will denote their transpose. We use $[\mathbf{A}|\mathbf{B}]$ for the horizontal concatenation of matrices, and $[\mathbf{A};\mathbf{B}] = \left[\mathbf{A^t}|\mathbf{B^t}\right]^t$ for the vertical concatenation. Unless otherwise stated, the norm $\|\cdot\|$ considered here is the ℓ_2 norm. We denote by $\langle \mathbf{x}, \mathbf{y} \rangle$ the canonical inner product between vectors $\mathbf{x}$ and $\mathbf{y}$, and by $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ their distance.

Cryptographic Primitives.

Definition 1 (Commitments). *A* non-interactive labelled commitment scheme $\mathcal{C}$ *is defined by three algorithms:*

- $\mathsf{SetupCom}(1^\kappa)$ *takes as input the security parameter κ and outputs the global parameters, passed through the CRS ρ to all other algorithms;*
- $\mathsf{Com}^\ell(x)$ *takes as input a label ℓ and a message x, and outputs a pair (C, δ), where C is the commitment of x for the label ℓ, and δ is the corresponding opening data (a.k.a. decommitment information). This is a probabilistic algorithm.*
- $\mathsf{VerCom}^\ell(C, x, \delta)$ *takes as input a commitment C, a label ℓ, a message x, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, x and ℓ. It always outputs 0 (false) on $x = \perp$.*

The basic properties required for commitments are *correctness* (for all correctly generated CRS ρ, all commitments and opening data honestly generated pass the verification VerCom test), the *hiding property* (the commitment does not leak any information about the committed value) and the *binding property* (no adversary can open a commitment in two different ways).

A commitment scheme is said *equivocable* if it has a second setup $\mathsf{SetupComT}(1^\kappa)$ that additionally outputs a trapdoor τ, and two algorithms

- $\mathsf{SimCom}^\ell(\tau)$ that takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
- $\mathsf{OpenCom}^\ell(\mathsf{eqk}, C, x)$ that takes as input a commitment C, a label ℓ, a message x, an equivocation key eqk, and outputs an opening data δ for C and ℓ on x.

Finally, a commitment scheme $\mathcal{C}$ is said *extractable* if it has a second setup algorithm $\mathsf{SetupComT}(1^\kappa)$ that additionally outputs a trapdoor τ, and a new algorithm

- $\mathsf{ExtCom}^\ell(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

We give an informal overview of the useful properties in Appendix A.1 in order to help the unfamiliar reader (formal definitions and results can be found in [ABB$^+$13]).

[1] By "fixed function", we mean that $f = \omega(\sqrt{\log n})$ always refers to the very same function.

Definition 2 (Verifiable Chameleon Hash).
A verifiable Chameleon Hash Function is defined by five algorithms $\mathsf{CH} = (\mathsf{KeyGen}, \mathsf{VKeyGen}, \mathsf{CH}, \mathsf{Valid}, \mathsf{Coll})$*:*

- $\mathsf{KeyGen}(\kappa)$*: Outputs the chameleon hash key* ck *and the trapdoor* tk*;*
- $\mathsf{VKeyGen}(\mathsf{ck})$*: Outputs the chameleon designated verification key* vk *and the trapdoor* vtk*. This trapdoor can be empty or public if the chameleon hash is publicly verifiable.*
- $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, m; r)$*: Picks a random r, and outputs the chameleon hash a as well as the* witness d, i.e. *the corresponding data needed to verify a.*
- $\mathsf{Valid}(\mathsf{ck}, \mathsf{vk}, m, a, d, \mathsf{vtk})$*: Allows to check that the sender knows how to open a Chameleon Hash a to a specific value m for the witness d. The verification can be public if* vtk *is empty or public, or specific to the receiver otherwise.*
- $\mathsf{Coll}(\mathsf{ck}, \mathsf{vk}, m, r, m', \mathsf{tk})$*: Takes as input the public keys, the trapdoor* tk*, a start message m and randomness r and a target message m' and outputs a target randomness r' such that if* $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, m; r) = (a, d)$*, then* $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, m'; r') = (a, d')$.

The security notions needed for verifiable CH are collision resistance, uniformity and soundness for the verification (see Appendix A.3 for details).

Definition 3 (Digital Signature Scheme [DH76,GMR88]).
A digital signature scheme allows a signer to produce a verifiable proof that he indeed produced a message. It is described through four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\kappa)$ *where κ is the security parameter, generates the global parameters* param *of the scheme, for example the message space;*
- $\mathsf{KeyGen}(\mathsf{param})$*, outputs a pair of* $(\mathsf{sk}, \mathsf{vk})$*, where* sk *is the (secret) signing key, and* vk *is the (public) verification key;*
- $\mathsf{Sign}(\mathsf{sk}, M; \mu)$*, outputs a signature $\sigma(M)$, on a message M, under the signing key* sk*, and some randomness μ;*
- $\mathsf{Verify}(\mathsf{vk}, M, \sigma)$ *checks the validity of the signature σ with respect to the message M and the verification key* vk*. And so outputs a bit.*

In the following we expect the scheme to fulfill correctness and strong one-time unforgeability under chosen message attacks. We recall this property and the transformation from chameleon hash to strong one-time signature in Appendix A.4.

Definition 4 (Labelled Encryption Scheme). *A labelled public-key encryption scheme $\mathcal{E}$ over a set of messages $\mathcal{M}$ is defined by four algorithms:*

- $\mathsf{Setup}(1^\kappa)$*, where κ is the security parameter, generates the global parameters* param *of the scheme;*
- $\mathsf{KeyGen}(\mathsf{param})$ *generates a pair of keys, the public encryption key* ek *and the private decryption key* dk*;*
- $\mathsf{Encrypt}^{\ell}(\mathsf{ek}, m; r)$ *produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key* ek*, using the random coins r;*
- $\mathsf{Decrypt}^{\ell}(\mathsf{dk}, c)$ *outputs the plaintext m encrypted in c under the label ℓ, or $\perp$ for an invalid ciphertext.*

and satisfies the following property:

– Correctness*: for all security parameter κ, with overwhelming probability over the key pair* $(\mathsf{ek}, \mathsf{dk})$*, for any label ℓ, all random coins r and all messages m, we have* $\mathsf{Decrypt}^{\ell}(\mathsf{dk}, \mathsf{Encrypt}^{\ell}(\mathsf{ek}, m; r)) = m$.

The IND-CCA2 and tag-IND-CCA2 security notions are defined in Appendix A.5, as well as the method to convert a tag-IND-CCA2 encryption scheme into IND-CCA2.

Definition 5 (Approximate Smooth Projective Hash Functions [KV09]). *Denote $\widetilde{\mathfrak{L}} \subseteq \mathfrak{L} \subseteq X$ the languages of ciphertexts defined by*

$$\widetilde{\mathfrak{L}} = \{(\ell, C, M) | \exists \rho,\ C = \mathsf{Encrypt}(\mathsf{ek}, \ell, M; \rho)\}$$
$$\mathfrak{L} = \{(\ell, C, M) | \mathsf{Decrypt}(\mathsf{dk}, \ell, C) = M\}$$

where $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a labelled encryption scheme and the witness relation is implicitly defined if and only if* $C = \mathsf{Encrypt}(\mathsf{ek}, \ell, M; \rho)$*. An approximate smooth projective hash function (SPHF) for these languages is defined by four probabilistic polynomial-time algorithms:*

– $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$ *generates a hashing key* hk *for the language parameter* param*;*
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ *derives a projection key* hp *from the hashing key* hk*, the language parameter* param*, and the word W;*
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ *outputs a hash value $\mathscr{H} \in \{0,1\}^{\nu}$ (for some positive integer $\nu = \Omega(n)$) from the hashing key* hk*, for the word $W \in X$ and the language parameter* param*;*
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$ *outputs a projected hash value $\mathscr{H}' \in \{0,1\}^{\nu}$ from the projection key* hp*, and the witness w, for the word $W \in \widetilde{\mathfrak{L}}$ and the language parameter* param*;*

We describe in Appendix A.2 the properties of approximate correctness and smoothness, as well as how to obtain an approximate SPHF from an Approximate Bit-PHF.

Security Notions. Throughout this paper, we assume basic familiarity with the universal composability framework [Can01]. A quick introduction, as well as the ideal functionality for oblivious transfer are given in Appendix A.6.

Lattice-Based Assumptions.

LATTICES. An m-dimensional *lattice* Λ is a (non-zero) discrete subgroup of $\mathbb{R}^m$. A *basis* of Λ is a linearly-independent set of vectors whose $\mathbb{Z}-$span is Λ. Equivalently, if $\mathbf{B} \in \mathbb{R}^{m \times n}$ is a basis of Λ then we can write $\Lambda = \{\mathbf{Bs} \mid \mathbf{s} \in \mathbb{Z}^n\}$ where $n \leq m$. In this work, we are mostly concerned with full-rank integer lattices, *i.e.* $\Lambda \subseteq \mathbb{Z}^m$ with $n = m$.

We define the *dual lattice* of Λ as $\Lambda^* = \{\mathbf{x} \in \mathrm{Span}_{\mathbb{R}}(\Lambda) \mid \forall \mathbf{y} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$.

Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be arbitrary and define the following full-rank m-dimensional q-ary lattices:

$$\Lambda(\mathbf{A}) = \{\mathbf{As} \mid \mathbf{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m$$
$$\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{h} \in \mathbb{Z}^m \mid \mathbf{h}^t\mathbf{A} = \mathbf{0}^t \bmod q\}.$$

It is easy to see that up to a scaling factor, $\Lambda(\mathbf{A})$ and $\Lambda^{\perp}(\mathbf{A})$ are dual of each other: $\Lambda(\mathbf{A}) = q \cdot \Lambda^{\perp}(\mathbf{A})^{*}$. For any $\mathbf{u} \in \mathbb{Z}_q^n$ admitting an integral solution to $\mathbf{A}\mathbf{x} = \mathbf{u} \bmod q$, we define the coset of $\Lambda^{\perp}(\mathbf{A})$ as $\Lambda^{\perp}_{\mathbf{u}} = \{\mathbf{h} \in \mathbb{Z}^m \mid \mathbf{h}^t \mathbf{A} = \mathbf{u}^t \bmod q\} = \Lambda^{\perp}(\mathbf{A}) + \mathbf{x}$.

When there is no confusion about which matrix $\mathbf{A}$ is used, we will simply denote these lattices Λ, $\Lambda^{\perp}$, and $\Lambda^{\perp}_{\mathbf{u}}$ respectively.

GAUSSIANS. For $\sigma > 0$ and $\mathbf{c} \in \mathbb{R}^m$, we define the *Gaussian weight function* on $\mathbb{R}^m$:

$$\rho_{\sigma,\mathbf{c}} = \mathbf{x} \mapsto \exp\left(-\pi \|\mathbf{x} - \mathbf{c}\|^2 / \sigma^2\right).$$

Similarly, if Λ is an m-dimensional lattice, we define the discrete Gaussian distribution over Λ, of parameter s and centered in $\mathbf{c}$ by:

$$\forall \mathbf{x} \in \Lambda, D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\Lambda,\mathbf{c}}(\mathbf{x})}.$$

When $\mathbf{c} = 0$, we will simply write ρ_σ and $D_{\Lambda,\sigma}$.

An important quantity associated to a lattice is its *smoothing parameter*, introduced by Micciancio and Regev [MR04]: for a lattice Λ and any $\epsilon \in (0,1)$, the smoothing parameter of Λ, denoted $\eta_\epsilon(\Lambda)$, is defined as the smallest $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \mathbf{0}) \leq \epsilon$. In particular, we recall the bound of the smoothing parameter of $\mathbb{Z}^m$.

Lemma 1. *[BBDQ18, Lemma 2.5] For all integer $m > 1$, $\epsilon \in (0, 1/2)$, the smoothing parameter of $\mathbb{Z}^m$ satisfies $\eta_\epsilon(\mathbb{Z}^m) \leq C\sqrt{\log(m/\epsilon)}$ for some universal constant $C > 0$.*

CRYPTOGRAPHIC ASSUMPTIONS.

Definition 6 (Short Integer Solution (SIS)). *Let $q \geq 2$, $\beta > 0$, and some $m = \mathsf{poly}(n)$. The Short Integer Solution problem $\mathsf{SIS}_{q,\beta}$ consists in, given uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, finding a relatively short nonzero $\mathbf{z} \in \Lambda^{\perp}(\mathbf{A})$ such that:*

$$\mathbf{A}\mathbf{z} = 0 \bmod q \text{ and } \|\mathbf{z}\| \leq \beta.$$

In [Ajt96], Ajtai showed that for $q \geq \beta\sqrt{n} \cdot \omega(\sqrt{\log n})$, $\mathsf{SIS}_{q,\beta}$ is at least as hard as solving worst-case SIVP to within $\tilde{O}(\beta\sqrt{n})$ factors.

Definition 7 (Learning with Errors (LWE)). *Let $q \geq 2$ and χ be a distribution over $\mathbb{Z}$. The Learning with Errors problem $\mathsf{LWE}_{\chi,q}$ consists in, given polynomially many samples, distinguishing the two following distributions:*

- *$(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where $\mathbf{a}$ is uniform in $\mathbb{Z}_q^n$, $e \leftarrow \chi$, and $\mathbf{s} \in \mathbb{Z}_q^n$ is a fixed secret chosen uniformly,*
- *$(\mathbf{a}, b)$ where $\mathbf{a}$ is uniform in $\mathbb{Z}_q^n$, and b is uniform in $\mathbb{Z}_q$.*

In [Reg05], Regev showed that for $\chi = D_{\mathbb{Z},\sigma}$, for any $\sigma \geq 2\sqrt{n}$, and q such that $q/\sigma = \mathsf{poly}(n)$, $\mathsf{LWE}_{\chi,q}$ is at least as hard as solving worst-case SIVP for polynomial approximation factors. We recall that there are (quantum) reductions from SIS to LWE [Reg05,SSTX09] and that LWE is *at least as hard as* SIS.

TRAPDOOR FOR LWE. We present here the Micciancio-Peikert trapdoors introduced in [MP12], to build our public matrix $\mathbf{A}$. Let $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{A}\mathbf{s} + \mathbf{e}$, define the parity-check matrix $\mathbf{G}$ as $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n \times nk}$, where $\mathbf{g}^t = \left[1, 2, \ldots, 2^{k-1}\right]$ and $k = \lceil \log q \rceil$, and let $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ be invertible.

Lemma 2 ([BBDQ18, Lemma 2.8]). *There exists two PPT algorithms $\mathsf{TrapGen}$ and $g^{-1}_{(\cdot)}$ with the following properties assuming $q \geq 2$ and $m \geq \Theta(n \log q)$:*

- $\mathsf{TrapGen}\left(1^n, 1^m, q\right)$ *outputs* $(\mathbf{T}, \mathbf{A}_0)$*, where the distribution of the matrix* $\mathbf{A}_0$ *is at negligible distance from uniform in* $\mathbb{Z}_q^{m\times n}$*, and such that* $\mathbf{T}\mathbf{A}_0 = 0$*, where* $s_1(\mathbf{T}) \leq O\left(\sqrt{m}\right)$ *and where* $s_1(\mathbf{T})$ *is the operator norm of* $\mathbf{T}$*, which is defined as* $\max_{\mathbf{x}\neq 0} \|\mathbf{T}\mathbf{x}\| / \|\mathbf{x}\|$.
- *Let* $(\mathbf{T}, \mathbf{A}_0) \leftarrow \mathsf{TrapGen}\left(1^n, 1^m, q\right)$*. Let* $\mathbf{A}_\mathbf{H} = \mathbf{A}_0 + [\mathbf{0}; \mathbf{G}\mathbf{H}]$ *for some invertible matrix* $\mathbf{H}$ *called a* tag*. Then,* $\mathbf{T}\mathbf{A}_\mathbf{H} = \mathbf{G}\mathbf{H}$*. Furthermore, if* $\mathbf{x} \in \mathbb{Z}_q^m$ *can be written as* $\mathbf{A}_\mathbf{H}\mathbf{s} + \mathbf{e}$ *where* $\|\mathbf{e}\| \leq B' := q/\Theta\left(\sqrt{m}\right)$*, then* $g_{\mathbf{A}_\mathbf{H}}^{-1}(\mathbf{T}, \mathbf{x}, \mathbf{H})$ *outputs* $(\mathbf{s}, \mathbf{e})$.

More precisely, [MP12] describes two types of TrapGen instantiations:

- *Statistical instantiation.* We sample a uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{\bar{m}\times n}$ where $\bar{m} = m - nk = \Theta(n \log q)$, and some $\mathbf{R} \leftarrow \mathcal{D}^{nk\times\bar{m}}$, where the distribution $\mathcal{D}^{nk\times\bar{m}}$ assigns probability $1/2$ to 0, and $1/4$ to ± 1.
- *Computational instantiation.* We sample a uniform $\hat{\mathbf{A}} \in \mathbb{Z}_q^{n\times n}$ and let $\bar{\mathbf{A}} = \left[\mathbf{I}; \hat{\mathbf{A}}\right] \in \mathbb{Z}_q^{\bar{m}\times n}$ where $\bar{m} = 2n$, and some $\mathbf{R} \leftarrow D_{\mathbb{Z},\sigma}^{nk\times\bar{m}}$ for some $\sigma = \alpha q$, where $\alpha > 0$ is an LWE relative error rate.

In both instantiations, we output $\mathbf{T} = [-\mathbf{R}|\mathbf{I}_{nk}]$ along with $\mathbf{A}_0 = \left[\bar{\mathbf{A}}; \mathbf{R}\bar{\mathbf{A}}\right]$. Then, given a tag $\mathbf{H}$, we have $\mathbf{T}\left(\mathbf{A}_0 + [\mathbf{0}; \mathbf{G}\mathbf{H}]\right) = \mathbf{G}\mathbf{H}$. Furthermore, the authors of [MP12] show how to use a trapdoor for efficient Gaussian pre-image sampling for q-ary lattices Λ, denoted SampleD (see [MP12] and references therein).

3 Lattice-based Building Blocks

Chameleon Hash. We present here a Chameleon Hash constructed from the SIS assumption, following the chameleon hash given in [CHKP10] but using the Micciancio-Peikert trapdoor generation [MP12]. We here only present the scheme, since the security proof comes directly following the proof of Lemma 4.1 in [CHKP10]. We consider message $\mathbf{m} \in \{0,1\}^\ell$.

Let $k = \lceil \log q \rceil$ and $m = O(nk)$. Let $\mathcal{D} = D_{\mathbb{Z}^{\bar{m}\times nk}, \omega(\sqrt{\log n})}$ be the Gaussian distribution over $\mathbb{Z}^{\bar{m}\times nk}$ with parameter $\omega(\sqrt{\log n})$ and let $\sigma = O(\sqrt{nk})$ be a Gaussian parameter. Let the randomness space be defined as $\mathcal{R} = D_{\mathbb{Z}^m, \sigma\cdot\omega(\sqrt{\log n})}$. Then, the Chameleon Hash is defined as follows:

- $\mathsf{KeyGen}(\kappa)$: choose a random matrix $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times\ell}$.
  Sample $(\mathbf{R}_1, \mathbf{A}_1) \xleftarrow{\$} \mathsf{TrapGen}(1^n, 1^m, q)$. Define $\mathsf{ck} = (\mathbf{A}_0, \mathbf{A}_1)$ and $\mathsf{tk} = \mathbf{R}_1$.
- $\mathsf{VKeyGen}(\mathsf{ck})$: Outputs $\mathsf{vk} = \perp$, $\mathsf{vtk} = \perp$.
- $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, \mathbf{m}; \mathbf{r})$: choose a vector $\mathbf{r}$ from the randomness space $\mathcal{R}$: $\mathbf{r} \leftarrow D_{\mathbb{Z}^m, \sigma\cdot\omega(\sqrt{\log n})}$. Compute the chameleon hash value $\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{A}_1\mathbf{r}$. Return the chameleon hash $\mathbf{c}$ and the opening information $\mathbf{r}$ (which we will later commit using a CCA2 scheme).
- $\mathsf{Coll}(\mathsf{tk}, (\mathbf{m}_0, \mathbf{r}_0), \mathbf{m}_1)$: compute $\mathbf{u} = (\mathbf{A}_0\mathbf{m}_0 + \mathbf{A}_1\mathbf{r}_0) - \mathbf{A}_0\mathbf{m}_1$ and sample $\mathbf{r}_1 \in \mathbb{Z}^m$ according to $D_{\Lambda_\mathbf{u}^\perp(\mathbf{A}_1), \sigma\cdot\omega(\sqrt{\log n})}$: $\mathbf{r}_1 \xleftarrow{\$} \mathsf{SampleD}(\mathbf{R}_1, \mathbf{A}_1, \mathbf{u}, \sigma)$.
- $\mathsf{Verify}(\mathsf{ck}, \mathsf{vtk}, \mathbf{m}, \mathbf{c}, \mathbf{r})$: accept if $\|\mathbf{r}\| \leq \sigma\cdot\omega(\sqrt{\log n})\cdot\sqrt{m}$ and $\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{A}_1\mathbf{r}$; otherwise, reject.

It should be noted, that the trapdoor allows to recover not only a collision, but also a preimage if need be.

Labelled TAG-IND-CCA2 Encryption Scheme. Following A.5, in order to construct an IND-CCA2 labelled encryption scheme for messages of $2K$ bits, one simply has to use a TAG-IND-CCA2 scheme for bits, use the same label in all the encryptions, and then add a one-time signature, built for example by using the previous chameleon hash. Two candidate encryption schemes have been proposed in [KV09] and [BBDQ18]. In order to avoid a blow-up in parameters and running time, we use the latter one, that we recall here.

For this scheme, we assume q to be an odd prime. We set an encoding function for messages $\mathsf{Encode}(\mu \in \{0,1\}) = \mu \cdot (0, \ldots 0, \lceil q/2 \rceil)^t \in \mathbb{Z}_q^m$. Note that $2 \cdot \mathsf{Encode}(\mu) = (0, \ldots, 0, \mu)^t \bmod q$.

Let $\mathcal{R}$ be a ring with a subset $\mathcal{U} \subset \mathcal{R}^\times$ of invertible elements, of size 2^n, and with the *unit differences* property: if $u_1 \neq u_2 \in \mathcal{U}$, then $u_1 - u_2$ is invertible in $\mathcal{R}$. Let h be an injective ring homomorphism from $\mathcal{R}$ to $\mathbb{Z}_q^{n \times n}$ (see [MP12, Section 6.1 and 6.3] for an explicit construction). Note that if $u_1 \neq u_2 \in \mathcal{U}$, then $h(u_1 - u_2)$ is invertible, and thus an appropriate tag $H = h(u_1 - u_2)$ for the trapdoor.

Let $(\mathbf{T}, \mathbf{A}_0) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$. The public encryption key is $\mathsf{ek} = \mathbf{A}_0$, and the secret decryption key is $\mathsf{dk} = \mathbf{T}$. For all bits $\mu[i], i \in |\mu|$:

– $\mathsf{Encrypt}(\mathsf{ek} = \mathbf{A}_0,\ u \in \mathcal{U},\ \mu[i] \in \{0,1\})$ encrypts the message μ under the public key ek and for the tag u, as follows: Let $\mathbf{A}_u = \mathbf{A}_0 + [\mathbf{0}\,;\, \mathbf{G}h(u)]$. Pick $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},t}^m$ where $t = \sigma\sqrt{m} \cdot \omega(\sqrt{\log n})$. Restart if $\|\mathbf{e}\| > B$, where $B \stackrel{\text{def}}{=} 2t\sqrt{m}$.[2] Output the ciphertext $\mathbf{c} = \mathbf{A}_u\mathbf{s} + \mathbf{e} + \mathsf{Encode}(\mu) \bmod q$.

– $\mathsf{Decrypt}(\mathsf{dk} = \mathbf{T},\ u \in \mathcal{U},\ \mathbf{c} \in \mathbb{Z}_q^m)$ decrypts the ciphertext $\mathbf{c}$ for the tag u using the decryption key dk as follows: Output

$$\begin{cases} \mu & \text{if } g_{\mathbf{A}_u}^{-1}(\mathbf{T}, 2\mathbf{c}, h(u)) = 2\mathbf{e} + (0, \ldots, 0, \mu) \text{ where } \mathbf{e} \in \mathbb{Z}^m \text{ and } \|\mathbf{e}\| \leq B' \\ \perp & \text{otherwise.}^3 \end{cases},$$

Since $\lceil q/2 \rceil$ is the inverse of $2 \bmod q$, we have

$$\mu' \stackrel{\text{def}}{=} \mathsf{Decrypt}(\mathbf{T}, u, \mathbf{c}) \neq \perp \quad \Longleftrightarrow \quad d(\mathbf{c} - \mathsf{Encode}(\mu'), \Lambda(\mathbf{A}_u)) < B' \ .$$

Suppose that $m \geq \Theta(n \log q)$. Note that $d(\mathsf{Encode}(1), \Lambda(\mathbf{A}_u)) > B'$ simultaneously for all u with overwhelming probability over the randomness of $\mathsf{TrapGen}$ (using a union bound, as in [GPV08, Lemma 5.3] for instance). Then the scheme is correct as long as $B \leq B'$, or equivalently $\sigma m^{3/2} \cdot \omega(\sqrt{\log n}) \leq q$.

Theorem 1. *Assume $m \geq \Theta(n \log q)$. The above scheme is tag-IND-CCA2 assuming the hardness of the LWE$_{\chi,q}$ problem for $\chi = D_{\mathbb{Z},\sigma}$.*

Smooth Projective Hash Function. A natural approach to define an approximate bit-PHF on the former encryption scheme E is the following, given in [BBDQ18]:

– $\mathsf{HashKG}(\mathfrak{L}_E, \mathbf{A})$ outputs $\mathsf{hk} = \mathbf{h} \leftarrow D_{\mathbb{Z},s}^m$;
– $\mathsf{ProjKG}(\mathbf{h}, (\mathfrak{L}_E, \mathbf{A}))$ outputs $\mathsf{hp} = \mathbf{p} = \mathbf{A}^t\mathbf{h}$;
– $\mathsf{Hash}(\mathbf{h}, (\mathfrak{L}_E, \mathbf{A}), \mathbf{c})$ outputs $\mathscr{H} = R(\langle \mathbf{h}, \mathbf{c} \rangle)$;

[2] This happens only with exponentially small probability $2^{-\Theta(n)}$.

[3] Note that the inversion algorithm $g_{(\cdot)}^{-1}$ can succeed even if $\|\mathbf{e}\| > B'$, depending on the randomness of the trapdoor. It is crucial to reject decryption nevertheless when $\|\mathbf{e}\| > B'$ to ensure CCA2 security. We also recall that $B' \stackrel{\text{def}}{=} q/\Theta(\sqrt{m})$.

– $\mathsf{ProjHash}(\mathbf{p}, (\mathfrak{L}_E, \mathbf{A}), \mathbf{c}, (\mathbf{s}, \mathbf{e}))$ outputs $\mathscr{H}' = R(\langle \mathbf{p}, \mathbf{s} \rangle)$;

where R is a *rounding* function to be chosen later and $s > 0$ is a parameter to be chosen later too. Let $r^\sharp, r^\flat$ be two q-periodic defined on $[-q/2, q/2]$ by:

$$\forall x \in \left[\frac{-q}{2}, \frac{q}{2}\right], \quad r^\sharp(x) = \begin{cases} 1 \text{ if } |x| \in [-q/4, q/4), \\ 0 \text{ otherwise,} \end{cases} \quad \text{and} \quad r^\flat(x) = \begin{cases} \frac{1}{2T} \text{ if } |x| \leq T, \\ 0 \text{ otherwise.} \end{cases}$$

We denote $\odot$ the convolution of q-periodic functions. The following theorem guarantees the statistical correctness of the construction when we instantiate it with a well-defined rounding function R.

Theorem 2. *Suppose $q = O(2^n)$ is superpolynomial in n, $m = \Theta(n \log q)$. Set parameters:*

1. *T such that T/q and q/T^2 are both negligible in n (using $T = q^{2/3}$ for instance),*
2. *$k = \Theta(n)$, and*
3. *$s \geq \Theta(\sqrt{n})$ such that $s/q = \mathsf{negl}(n)$ and $s = \Omega(\frac{mk^2q^2}{T^2})$, which exists by construction of T.*

Define a probabilistic rounding function $R : \mathbb{Z}_q \to \{0, 1\}$ such that $\Pr[R(x) = 1] = (r^\sharp \odot r^\flat)(x)$. Then the bit-PHF achieves $(1/3 + o(1))$-universality and statistical correctness.

Proof. The theorem follows from [BBDQ18] using:

1. $N = kq/T$ (in which case NC^m is negligible in n), and
2. $\delta = \frac{q\sqrt{m}}{s}$.

APPROXIMATE SPHF BASED ON LWE. First, we instantiate the described approximate bit-PHF scheme with a concrete rounding function $R(x) = \frac{1}{2} + \frac{\cos(2\pi x/q)}{2}$ and the parameter $s = \Theta(m)$. Then the approximate SPHF is defined as follows:

– $\mathsf{HashKG}(\mathfrak{L}_E, \mathbf{A})$ generates a hashing key $\mathsf{hk} = (\mathsf{hk}_1, \ldots, \mathsf{hk}_\nu)$ by sampling ν times $\mathsf{hk}_i \xleftarrow{\$} D^m_{\mathbb{Z},s}$, where $\nu = \Omega(\kappa)$.
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}_E, \mathbf{A}))$ derives a projection key hp from the hashing key hk, by computing $\mathsf{hp}_i = \mathbf{A}^t\mathsf{hk}_i$ (for $i \in [\![1, \nu]\!]$) and setting $\mathsf{hp} = (\mathsf{hp}_1, \ldots, \mathsf{hp}_\nu)$.
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}_E, \mathbf{A}), \mathbf{c})$ outputs a hash value $\mathscr{H} \in \{0, 1\}^\nu$, by computing the various hash values $\mathscr{H}_i = R(\langle \mathsf{hk}_i, \mathbf{c} \rangle)$ (for $i \in [\![1, \nu]\!]$) and concatenating the outputs: $\mathscr{H} = \mathscr{H}_1 \| \ldots \| \mathscr{H}_\nu$.
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}_E, \mathbf{A}), \mathbf{c}, (\mathbf{s}, \mathbf{e}))$ outputs a projected hash value $\mathscr{H}' \in \{0, 1\}^\nu$, by computing the projected hash values $\mathscr{H}'_i = R(\langle \mathsf{hk}_i, \mathbf{s} \rangle)$ (for $i \in [\![1, \nu]\!]$) and concatenating them: $\mathscr{H}' = \mathscr{H}'_1 \| \ldots \| \mathscr{H}'_\nu$.

4 SPHF Friendly Commitment based on LWE

Having recalled in the former sections the necessary building blocks based on lattice assumptions, we now describe how to construct the main ingredient for oblivious transfer, which is SPHF-friendly commitment schemes, following the methodology given in [ABB+13] and [BC15] and recalled in Appendix B for completeness. The correctness and security directly follow from the generic theorem [BC15] since the chameleon hash is publicly verifiable and collision-resistant and the encryption scheme is IND-CCA2 secure and accepts an SPHF on the language of valid ciphertexts.

In our construction, we select the parameters q and n in order to achieve the desired level of security for the LWE-based schemes, and the remaining parameters are instantiated using the computational instantiation of the trapdoor with $m = n(k+2)$. Let σ be the Gaussian parameter for the trapdoor sampling $\mathsf{TrapGen}$. Furthermore, we denote $t = |N|$ where N is the number of lines of the database.

Let $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ and $\mathcal{C} = (\mathsf{KeyGen}, \mathsf{CH}, \mathsf{Coll}, \mathsf{Verify})$ denote the LWE-based instantiations of the TAG-IND-CCA2 encryption scheme and the Chameleon Hash described in Section 3, respectively.

The concrete construction for the SPHF Friendly Commitment is given in Figure 1.

- **Setup:** $\mathsf{SetupComT}(1^\kappa)$
  - Generate a public key ek for the encryption scheme: $\mathsf{ek} = \mathbf{A}^{\mathcal{E}} \overset{\$}{\leftarrow} \mathcal{E}.\mathsf{KeyGen}(1^\kappa)$.
  - Generate a hash key ck for the chameleon hash scheme: $\mathsf{ck} = (\mathbf{A}_0^{\mathcal{C}}, \mathbf{A}_1^{\mathcal{C}}) \overset{\$}{\leftarrow} \mathcal{C}.\mathsf{KeyGen}(1^\kappa)$.
  - Generate various parameters param for $\mathcal{E}$ (for example, the subset $\mathcal{U}$ with the *unit differences* property): $\mathsf{param} \leftarrow \mathcal{E}.\mathsf{Setup}(1^\kappa)$.
  - Output $(\mathsf{ek}, \mathsf{ck}, \mathsf{param})$.
- **Commit:** $\mathsf{Com}^\ell(\mathbf{m})$, for $\mathbf{m} = (m_i) \in \{0,1\}^t$ and a label ℓ.
  - For each $i \in [\![1,t]\!]$, commit to m_i to get a hash value $\mathbf{a}_i \in \mathbb{Z}^m$ and an opening information $\mathbf{d}_{i,m_i} \in \mathbb{Z}^m$: $(\mathbf{a}_i, \mathbf{d}_{i,m_i}) \overset{\$}{\leftarrow} \mathcal{C}.\mathsf{CH}(\mathsf{ck}, m_i)$. Then sample at random $\mathbf{d}_{i,1-m_i} \overset{\$}{\leftarrow} D_{\mathbb{Z}^m, \sigma\cdot\omega(\sqrt{\log n})}$. We denote as $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_t)$ and $\mathbf{d}$ the concatenation of $(\mathbf{d}_{i,j})_{i\in[\![1,t]\!], j\in\{0,1\}}$.
  - Encrypt the opening information $\mathbf{d}$ as follows. We suppose that each entry of $\mathbf{d}$ can be represented as a ν-bit string $(d^\mu)_{\mu\in[\![1,\nu]\!]}$ without loss of generality.
    (a) Generate a signature key sk and an associated verification key vk: run $\mathcal{C}.\mathsf{KeyGen}$ twice, and get $\mathsf{ck}_i, \mathsf{tk}_i$ for $i \in \{0,1\}$. Set $(\mathbf{z}_0, \mathbf{r}_0) \overset{\$}{\leftarrow} \mathcal{C}.\mathsf{CH}(\mathsf{ck}_0, 0), (\mathbf{z}_1, \mathbf{r}_1) \overset{\$}{\leftarrow} \mathcal{C}.\mathsf{CH}(\mathsf{ck}_1, z_0))$. Set $\mathsf{vk} = (\mathsf{ck}_0, \mathsf{ck}_1, \mathbf{z}_1)$, $\mathsf{sk} = (\mathsf{tk}_0, \mathsf{tk}_1, \mathbf{r}_0, \mathbf{r}_1, \mathbf{z}_0)$.
    (b) Map the verification key vk to a random element of the set $\mathcal{U}$ of $\mathcal{E}$ by using an injective map $\mathcal{H}$.
    (c) For each $i \in [\![1,t]\!]$, $j \in \{0,1\}$ and $\mu \in [\![1,\nu]\!]$, compute $\mathbf{c}_{i,j}$ to be the concatenation of $\mathbf{c}_\mu \overset{\$}{\leftarrow} \mathcal{E}.\mathsf{Encrypt}(\mathsf{ek}, \mathcal{H}(\mathsf{vk}), \mathbf{d}_{i,j}^\mu)$ and set $\mathbf{s}_{i,j}$ to be the concatenation of all sampled secret $\mathbf{s}_{i,j}^\mu \in \mathbb{Z}^m$.
    (d) Sign $(\mathbf{c}_1, \ldots, \mathbf{c}_{\bar{t}}, \mathsf{vk}, \ell)$ under the secret key sk: get $\mathbf{r}_0' \overset{\$}{\leftarrow} \mathcal{C}.\mathsf{Coll}(\mathsf{tk}_0, 0, \mathbf{r}_0, (\mathbf{c}_1, \ldots, \mathbf{c}_{\bar{t}}, \mathsf{vk}, \ell))$ where $\bar{t} = 2t$.
    (e) Set $\mathbf{b} = (\mathbf{c}_1, \ldots, \mathbf{c}_{\bar{t}}, \mathsf{vk}, [\mathbf{r}_0|\mathbf{r}_0'])$ and $\mathbf{s} = (\mathbf{s}_{i,j})_{i\in[\![1,t]\!], j\in\{0,1\}}$.
  - Output the commitment $C = (\mathbf{a}, \mathbf{b})$, and the opening information $\delta = (\mathbf{s}_{1,m_1}, \ldots, \mathbf{s}_{t,m_t})$.

Fig. 1. SPHF Friendly Commitment based on LWE

5 Oblivious Transfer based on LWE

Notations and Building Blocks. We denote by $\mathsf{DB} = (\mathsf{DB}_1, \ldots, \mathsf{DB}_N)$ the database of the server containing $N = 2^t$ lines, and j the line requested by the user in an oblivious way. We assume the existence of

- a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database: None of the PRGs suggested for cryptographic use would be affected by quantum computers, other than perhaps the $O(\sqrt{n})$ factor incurred by Grover's algorithm. Thus, we can simply use AES in counter mode as a PRG, taking advantage of AES-NI instructions when available for the implementation. As a general rule, for 128 bits of post-quantum security, one can safely use key sizes of 256 bits.
- an IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter: We can take the constructions from the literature (for example, [NAB$^+$17], etc.), or we can reuse the construction of the TAG-IND-CCA2 scheme described in Section 3 with a fixed tag (for example, the tag $\mathbf{H} = \mathbf{I}$), then the resulting construction will be IND-CPA secure.
- an SPHF-friendly commitment scheme: We construct it as described in Section 4 from the compatible CCA-encryption and chameleon hash from Section 3 (the language $\mathfrak{L}$ is defined from the encryption scheme as in Definition 5).

Construction. We follow the generic construction proposed in [ABB$^+$13,BC15], giving the protocol presented on Figure 2, instantiating the primitives needed by the building blocks based on LWE described above. Since the commitment scheme is constructed from a secure publicly-verifiable chameleon hash and a secure CCA encryption scheme admitting an SPHF on the language of valid ciphertexts, as described in Section 3, the proof of the following security theorem thus directly follows from the theorem of the generic construction given in [BC15].

Theorem 3. *The oblivious transfer scheme described in Figure 2 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels, under LWE assumption.*

Acknowledgments

We would like to thank the anonymous reviewers for detailed comments. This work was supported in part by the French ANR projects IDFIX (ANR-16-CE39-0004) and CryptiQ (ANR-18-CE39-0015).

References

[ABB$^+$13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Heidelberg, December 2013.

CRS: $\rho = (\mathsf{ek}, \mathsf{ck}, \mathsf{param}) \xleftarrow{\$} \mathsf{SetupCom}(1^\kappa)$, $\mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^\kappa)$.

Pre-flow:
1. Server generates a key pair $(\mathsf{ek}_{\mathrm{cpa}}, \mathsf{dk}_{\mathrm{cpa}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores $\mathsf{dk}_{\mathrm{cpa}}$ and completely erases the random coins used by KeyGen
2. Server generates a verification key pair $(\mathsf{vk}, \mathsf{vtk}) \xleftarrow{\$} \mathsf{VKeyGen}(\mathsf{ck})$ for CH, stores vtk and completely erases the random coins used by $\mathsf{VKeyGen}$
3. Server sends $\mathsf{ek}_{\mathrm{cpa}}$ and vk to User

Index query on j:
1. User chooses a random value J, computes $R \leftarrow F(J)$ and encrypts J under $\mathsf{ek}_{\mathrm{cpa}}$: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{ek}_{\mathrm{cpa}}, J)$
2. User computes $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(j)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
3. User stores δ, R and completely erases J and the random coins used by Com and $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends C and c to Server

Database input $(\mathsf{DB}_1, \ldots, \mathsf{DB}_N)$**:**
1. Server decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{dk}_{\mathrm{cpa}}, c)$ and then $R \leftarrow F(J)$
2. For $s = 1, \ldots, N$: Server computes $\mathsf{hk}_s \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, $\mathsf{hp}_s \leftarrow \mathsf{ProjKG}(\mathsf{hk}_s, (\mathfrak{L}, \mathsf{param}), (\ell, C))$, $K_s \leftarrow \mathsf{Hash}(\mathsf{hk}_s, (\mathfrak{L}, \mathsf{param}), (\ell, C))$, and $\mathsf{db}_s \leftarrow R \oplus K_s \oplus \mathsf{DB}_s$
3. Server erases everything except $(\mathsf{hp}_s \mathsf{db}_s)_{s=1,\ldots,N}$ and sends them over the secure channel

Data recovery:
Upon receiving $(\mathsf{hp}_s, \mathsf{db}_s)_{s=1,\ldots,N}$, User computes $K_j \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathfrak{L}, \mathsf{param}), (\ell, C), \delta)$ and gets $\mathsf{DB}_j \leftarrow R \oplus K_j \oplus \mathsf{db}_j$.

Fig. 2. UC-Secure 1-out-of-t OT from an SPHF-Friendly Commitment (for Adaptive Security)

[ACFP14] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014. `http://eprint.iacr.org/2014/1018`.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, August 2009.

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, April / May 2017.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[BBDQ18] Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 644–674. Springer, Heidelberg, March 2018.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 65–86. Springer, Heidelberg, June 2015.

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Heidelberg, December 2016.

[BCG16] Olivier Blazy, Céline Chevalier, and Paul Germouty. Adaptive oblivious transfer and generalization. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 217–247. Springer, Heidelberg, December 2016.

[BCG17] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost optimal oblivious transfer from QA-NIZK. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 579–598. Springer, Heidelberg, July 2017.

[BFRLS18] Pauline Bert, Pierre-Alain Fouque, Adeline Roux-Langlois, and Mohamed Sabt. Practical implementation of ring-SIS/LWE based signature and IBE. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 271–291. Springer, Heidelberg, 2018.

[BG14] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796. ACM Press, October 2012.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

[CHK$^+$05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, Heidelberg, February / March 2013.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

[DDN03] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM review*, 45(4):727–784, 2003.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42. Springer, Heidelberg, August 2015.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[GM18] Nicholas Genise and Daniele Micciancio. Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 174–203. Springer, Heidelberg, April / May 2018.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[Gt18] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.1.2 edition, 2018. `http://gmplib.org/`.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, Heidelberg, August 2007.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, Heidelberg, December 2009.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

[Moh11] Payman Mohassel. One-time signatures and chameleon hash functions. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 302–319. Springer, Heidelberg, August 2011.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[NAB$^+$17] Michael Naehrig, Erdem Alkim, Joppe Bos, Leo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. Frodokem. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Sho01] Victor Shoup. Ntl: A library for doing number theory. 2001. `http://www.shoup.net/ntl/`.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, December 2009.

[WHC$^+$14] Xiao Shaun Wang, Yan Huang, T.-H. Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: Oblivious RAM for secure computation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 191–202. ACM Press, November 2014.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Details and Formal Definitions

A.1 Commitments

A commitment scheme is said *equivocable* if the following properties are satisfied: *trapdoor correctness* (all simulated commitments can be opened on any message), *setup indistinguishability* (one cannot distinguish the CRS ρ generated by $\mathsf{SetupCom}$ from the one generated by $\mathsf{SetupComT}$) and *simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to fake commitments), denoting by SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$.

A commitment scheme $\mathcal{C}$ is said *extractable* if the following properties are satisfied: *trapdoor correctness* (all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C, \tau) = x$), *setup indistinguishability* (as above) and *binding extractability* (one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data to an input x while the commitment does not extract to x).

A.2 Smooth Projective Hash Functions

An approximate SPHF as defined in Section 5 should satisfy the following properties:

- **Approximate correctness.** For any $n \in \mathbb{N}$, with overwhelming probability over the randomness of Setup, for any $W \in \widetilde{\mathfrak{L}}$ (and associated witness w), the value $\mathscr{H}$ output by $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ is approximately determined by $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ relative to the Hamming metric. More precisely, writing $\mathsf{HW}(a, b)$ the Hamming distance between two strings $a, b \in \{0,1\}^{\nu}$, the SPHF is ε-correct, if:

$$\Pr_{\mathsf{hk}}\left[\mathsf{HW}(\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)) > \epsilon \cdot \nu\right] = \mathsf{negl}(n) \ ,$$

  where the probability is taken over the choice of $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{param})$ and the random coins of Hash and ProjHash.[4]
- **Smoothness.** For any $n \in \mathbb{N}$, with overwhelming probability over the randomness of Setup, for all $W \in X \setminus \mathfrak{L}$ the following distributions have statistical distance negligible in n:

$$\left\{(\mathsf{param}, W, \mathsf{hp}, \mathscr{H}) \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}),\ \mathscr{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \end{array}\right\} \ ,$$

$$\left\{(\mathsf{param}, W, \mathsf{hp}, \mathscr{H}) \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}),\ \mathscr{H} \leftarrow \{0,1\}^{\nu}, \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \end{array}\right\} \ .$$

We now recall how to obtain an approximate SPHF from an Approximate Bit-PHF. In the following, we are going to start from the [BBDQ18] bit-SPHF. In order to use it, as a proper SPHF, we need to increase the size of the output of the hash function, by sampling several independent hash keys hk, and concatenating the output of all the corresponding Hash results.

Lemma 3. *Let* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *be an ε-correct approximate bit-PHF. Then the SPHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *defined as follows is an* $(\varepsilon + \varepsilon')$*-correct approximate SPHF, for any constant* $\varepsilon' > 0$.

[4] Contrary to previously known SPHFs, some of our SPHFs have randomized algorithms Hash and ProjHash.

- $\mathsf{HashKG}(1^\kappa)$ *generates a hashing key* $\mathsf{hk} = (\mathsf{hk}_1, \ldots, \mathsf{hk}_\nu)$ *by running* ν *times* $\mathsf{HashKG}'(1^\kappa)$*, where* $\nu = \Omega(n)$*;*
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ *derives a projection key* hp *from the hashing key* hk*, by computing* $\mathsf{hp}_i = \mathsf{ProjKG}'(\mathsf{hk}_i, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ *(for* $i \in \{1, \ldots, \nu\}$*) and setting* $\mathsf{hp} = (\mathsf{hp}_1, \ldots, \mathsf{hp}_\nu)$.
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ *outputs a hash value* $\mathscr{H} \in \{0,1\}^\nu$*, by computing the various hash values* $\mathscr{H}_i = \mathsf{Hash}(\mathsf{hk}_i, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$ *(for* $i \in \{1, \ldots, \nu\}$*) and concatenating the outputs:* $\mathscr{H} = \mathscr{H}_1 \| \ldots \| \mathscr{H}_\nu$*;*
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W, w)$ *outputs a projected hash value* $\mathscr{H}' \in \{0,1\}^\nu$*, by computing the projected hash values* $\mathscr{H}'_i = \mathsf{ProjHash}'(\mathsf{hp}_i, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W, w)$ *(for* $i \in \{1, \ldots, \nu\}$*) and concatenating them:* $\mathscr{H}' = \mathscr{H}'_1 \| \ldots \| \mathscr{H}'_\nu$*;*

Proof. **Approximate correctness.** We have for every i:

$$\Pr_{\mathsf{hk}_i}[\mathsf{Hash}'(\mathsf{hk}_i, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) = \mathsf{ProjHash}'(\mathsf{hp}_i, \mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W, w)] \geq 1 - \epsilon \ .$$

Hence, the property on the concatenation, using the Hoeffding bound.
Smoothness. This follows from a classical hybrid argument by considering intermediate distributions Δ_i where the first i values $\mathscr{H}_i$ are random, and the others are honestly computed, as each SPHF is independent and smooth.

A.3 Chameleon Hash

A Chameleon Hash Function is traditionally defined by three algorithms $\mathsf{CH} = (\mathsf{KeyGen}, \mathsf{CH}, \mathsf{Coll})$:

- $\mathsf{KeyGen}(\kappa)$: Outputs the chameleon hash key ck and the trapdoor tk;
- $\mathsf{CH}(\mathsf{ck}, m; r)$: Picks a random r, and outputs the chameleon hash a.
- $\mathsf{Coll}(\mathsf{ck}, m, r, m', \mathsf{tk})$: Takes as input the trapdoor tk, a start message and randomness pair (m, r) and a target message m' and outputs a target randomness r' such that $\mathsf{CH}(\mathsf{ck}, m; r) = \mathsf{CH}(\mathsf{ck}, m'; r')$.

The standard security notion for CH is collision resistance, which means it is infeasible to find $(m_1, r_1), (m_2, r_2)$ such that $\mathsf{CH}(\mathsf{ck}, m_1, r_1) = \mathsf{CH}(\mathsf{ck}, m_2, r_2)$ and $m_1 \neq m_2$ given only the Chameleon hash key ck. Formally, CH is $(t, \varepsilon) - \mathsf{coll}$ if for the adversary $\mathcal{A}$ running in time at most t we have:

$$\Pr\begin{bmatrix}(\mathsf{ck}, \mathsf{tk}) \stackrel{\$}{\leftarrow} \mathsf{KeyGen}(\kappa); ((m_1, r_1), (m_2, r_2)) \stackrel{\$}{\leftarrow} \mathcal{A}(\mathsf{ck}) \\ \wedge \quad \mathsf{CH}(\mathsf{ck}, m_1; r_1) = \mathsf{CH}(\mathsf{ck}, m_2; r_2) \wedge m_1 \neq m_2\end{bmatrix} \leq \varepsilon.$$

However, any user in possession of the trapdoor tk is able to find a collision using Coll. Additionally, Chameleon Hash functions have the uniformity property, which means the hash value leaks nothing about the message input. Formally, for all pair of messages m_1 and m_2 and the randomly chosen r, the probability distributions of the random variables $\mathsf{CH}(\mathsf{ck}, m_1, r)$ and $\mathsf{CH}(\mathsf{ck}, m_2, r)$ are computationally indistinguishable.

We need here the hash value to be verifiable, so that we add two $\mathsf{VKeyGen}$ and Valid algorithms (executed by the receiver) and we modify the existing algorithms as described in Definition 2.

Once again, we expect the chameleon hash to be collision resistant on the first part of the output, which means it is infeasible to find $(m_1, r_1), (m_2, r_2)$ such that $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, m_1, r_1) = (a, d_1)$ and $\mathsf{CH}(\mathsf{ck}, m_2, r_2) = (a, d_2)$ and $m_1 \neq m_2$ given only the Chameleon public keys ck and vk.

We expect the verification to be sound, which means that, given a tuple (m, a, d) satisfying $\mathsf{Valid}(\mathsf{ck}, \mathsf{vk}, m, a, d, \mathsf{vtk})$, there always exists at least one tuple (r, d') such that $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, m; r) = (a, d')$.

A.4 One-Time Signatures

The properties of a digital signature scheme can be defined as follows:

- *Correctness*: For every pair $(\mathsf{vk}, \mathsf{sk})$ generated by KeyGen, for every message M, and for all randomness μ, we have $\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M; \mu)) = 1$.
- *Strong One-Time Unforgeability under Chosen Message Attacks*. Even after querying a valid signature σ on chosen messages M, an adversary should not be able to output a fresh valid signature (possibly on M). To formalize this notion, we define a signing oracle OSign:
  - $\mathsf{OSign}(\mathsf{vk}, m)$: This oracle outputs a signature on m valid under the verification key vk. The resulting pair (m, σ) is added to the signed pair set $\mathcal{S}'$.

```
$\mathsf{Exp}^{\mathsf{st-ot-uf}}_{\mathcal{S},\mathcal{A}}(\kappa)$
1.$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$
2.$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3.$(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{OSign}(\mathsf{vk}, \cdot))$
4.$b \leftarrow \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$
5.IF $(m^*, \sigma^*) \in \mathcal{S}'$ OR $\#(\mathcal{S}) > 1$ RETURN $0$
6.ELSE RETURN $b$
```

The probability of success against this game is denoted by

$$\mathsf{Succ}^{\mathsf{st-ot-uf}}_{\mathcal{S},\mathcal{A}}(\kappa) = \Pr[\mathsf{Exp}^{\mathsf{st-ot-uf}}_{\mathcal{S},\mathcal{A}}(\kappa) = 1]$$

$$\mathsf{Succ}^{\mathsf{st-ot-uf}}_{\mathcal{S}}(\kappa, t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{st-ot-uf}}_{\mathcal{S},\mathcal{A}}(\kappa).$$

Our scheme requires the use of Strong One-Time Signature. Such primitive can naturally be built from chameleon hashes as was shown in [Moh11]. We remind the transformation here. Given a Chameleon Hash (CH.KeyGen, CH.CH, CH.Coll), one can define a Strong One-Time Signature in the following way:

- Sign.KeyGen: Runs CH.KeyGen twice, and get $\mathsf{ck}_i, \mathsf{tk}_i$ for $i \in \{0, 1\}$. Samples random $r_i \in \{0, 1\}^\kappa$. Sets $z_0 = CH.\mathsf{CH}(\mathsf{ck}_0, 0, r_0), z_1 = CH.\mathsf{CH}(\mathsf{ck}_1, z_0, r_1))$. Set $\mathsf{vk} = (\mathsf{ck}_0, \mathsf{ck}_1, z_1), \mathsf{sk} = (\mathsf{tk}_0, \mathsf{tk}_1, r_0, r_1, z_0)$.
- $\mathsf{Sign.Sign}(\mathsf{sk}, m)$: Gets $r'_0 = CH.\mathsf{Coll}(\mathsf{tk}_0, 0, r_0, m)$ and publishes $\sigma = r'_0, r_1$.
- $\mathsf{Sign.Verify}(\mathsf{vk}, m, \sigma)$: Computes $z'_0 = CH.\mathsf{CH}(\mathsf{vk}_0, m, \sigma_0)$ and checks whether $\mathsf{vk}_2 = CH.\mathsf{CH}(\mathsf{vk}_1, z'_0, \sigma_1)$.

A.5 Labelled Encryption Schemes

Definition 8 (IND-CCA2 Security).
An encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* IND-CCA2 *if the advantage of any polynomial-time adversary* $\mathcal{A}$ *in distinguishing* $\mathsf{Exp}^{\mathsf{ind\text{-}cca}-0}_{\mathcal{E},\mathcal{A}}(\kappa)$ *from* $\mathsf{Exp}^{\mathsf{ind\text{-}cca}-1}_{\mathcal{E},\mathcal{A}}(\kappa)$ *is negligible in the security parameter* κ*, where the experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\kappa)$ *is depicted in Figure 3. The adversary* $\mathcal{A}$ *transfers some internal state* state *between the various calls* `FIND` *and* `GUESS`*, and makes use of the oracle* $\mathsf{ODecrypt}$*:*

– $\mathsf{ODecrypt}^{\ell}(c)$*: This oracle outputs the decryption of* c *under the label* ℓ *and the challenge decryption key* dk*. The input queries* (ℓ, c) *are added to the list* $\mathcal{CT}$*.*

Informally, this notion states that an adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and can ask several decryption of ciphertexts as long as they are not the challenge one.

```
$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\kappa)$
    $\mathsf{param} \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^\kappa)$
    $(\mathsf{ek}, \mathsf{dk}) \stackrel{\$}{\leftarrow} \mathsf{KeyGen}(\mathsf{param})$
    $(\ell^*, m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\texttt{FIND} : \mathsf{ek})$
    $c^* \leftarrow \mathsf{Encrypt}^{\ell^*}(\mathsf{ek}, m_b)$
    $b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathsf{state}, \texttt{GUESS} : c^*)$
    If $((\ell^*, c^*) \in \mathcal{CT})$
      Return $0$
    Else
      Return $b'$
```

Fig. 3. Security Experiment for IND-CCA2 Security

This IND-CCA2 notion can be relaxed into a weaker tag-IND-CCA2 security notion.

Definition 9 (Tag-IND-CCA2 Security). *An encryption scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* tag-CCA2-secure *if the advantage of any polynomial-time adversary* $\mathcal{A}$ *in distinguishing* $\mathsf{Exp}^{\mathsf{tag\text{-}cca}-0}_{\mathcal{E},\mathcal{A}}(\kappa)$ *from* $\mathsf{Exp}^{\mathsf{tag\text{-}cca}-1}_{\mathcal{E},\mathcal{A}}(\kappa)$ *is negligible in the security parameter* κ*, where the experiments* $\mathsf{Exp}^{\mathsf{tag\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\kappa)$ *are defined as the experiments* $\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\kappa)$ *depicted in Figure 3, except that:*

– *The line*
    If $((\ell^*, c^*) \in \mathcal{CT})$
  is replaced by
    If $((\ell^*, \cdot) \in \mathcal{CT})$

In other words, the adversary is not allowed to query the decryption oracle on a ciphertext with the same label ℓ (also called a tag and denoted $\mathfrak{u}$ *in this context) as the challenge one.*

– *In addition the adversary chooses the label ℓ^* before seeing* ek, *i.e. the two lines*

  $(\mathsf{ek}, \mathsf{dk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(\mathsf{param})$
  $(\ell^*, m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathit{FIND} : \mathsf{ek})$

  are replaced by

  $(\ell^*, \mathsf{state}_0) \overset{\$}{\leftarrow} \mathcal{A}(1^\kappa)$
  $(\mathsf{ek}, \mathsf{dk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(\mathsf{param})$
  $(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathsf{state}_0, \mathit{FIND} : \mathsf{ek})$

Finally, we recall that the weaker IND-CPA security notion is defined similarly as the IND-CCA2 or tag-IND-CCA2 security notion, except that the adversary is not given access to the decryption oracle $\mathsf{ODecrypt}$. If the tag of a tag-IND-CCA2 encryption scheme is fixed to some public constant, then the resulting scheme is IND-CPA.

We can convert a tag-IND-CCA2 encryption scheme $(\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ with message space $\{0,1\}$ and label (a.k.a., tag) space $\{0,1\}^\kappa$ into an IND-CCA2 encryption scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with message space $\{0,1\}^\nu$ (for some ν polynomial in κ) and label space $\{0,1\}^*$, using [DDN03].

Concretely, we suppose that we have a strongly unforgeable one-time signature scheme and we define:

- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, uses $\mathsf{Setup}'(1^\kappa)$ to generate the global parameters param of the scheme;
- $\mathsf{KeyGen}(1^\kappa)$ outputs $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}'(1^\kappa)$;
- $\mathsf{Encrypt}^\ell(\mathsf{ek}, m)$, with $\ell \in \{0,1\}^*$ and $m \in \{0,1\}^\nu$, generates a signature key sk_σ and an associated verification key vk_σ (for the strongly unforgeable one-time signature, we suppose that vk_σ can be represented as a κ-bit string without loss of generality), computes for $1 \le i \le \nu$, $c_i \leftarrow \mathsf{Encrypt}'^{\mathsf{vk}_\sigma}(\mathsf{ek}, m_i)$, and outputs $c \overset{\text{def}}{=} (c_1, \dots, c_\nu, \mathsf{vk}_\sigma, \sigma)$, where σ is a signature under sk_σ of $(c_1, \dots, c_\nu, \mathsf{vk}_\sigma, \ell)$;
- $\mathsf{Decrypt}^\ell(\mathsf{dk}, c)$, with $\ell \in \{0,1\}^*$, parses c as $(c_1, \dots, c_\nu, \mathsf{vk}_\sigma, \sigma)$, abort (i.e., return $\perp$) if σ is not a valid signature of $(c_1, \dots, c_\nu, \mathsf{vk}_\sigma, \ell)$ under vk_σ, otherwise computes for $1 \le i \le \nu$, $m_i = \mathsf{Decrypt}'^{\mathsf{vk}_\sigma}(\mathsf{dk}, c_i)$, and output the bit string $m \in \{0,1\}^\nu$ corresponding to the concatenation of $m_1, \dots, m_\nu$.

In the following, in order to supersede the decryption by an implicit decommitment, we require the encryption to admit an efficient implicit decommitment. We will call an SPHF-friendly encryption, an encryption where there exists an SPHF for the Language of valid ciphertexts of a message m using as sole witness the randomness used in the encryption.

A.6 Security Notions

UC Framework The goal of the UC framework [Can01] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol Π emulates $\mathcal{F}$, one has to construct, for any polynomial adversary $\mathcal{A}$ (which controls the communication between the players), a simulator $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathcal{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathcal{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, *i.e.* allowed to corrupt users whenever it likes to, or *static*, *i.e.* required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathcal{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf.

Simple UC Framework Canetti, Cohen and Lindell formalized a simpler variant in [CCL15], that we use here. This simplifies the description of the functionalities for the following reasons (in a nutshell): All channels are automatically assumed to be authenticated (as if we worked in the $\mathcal{F}_{\text{AUTH}}$-hybrid model); There is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party), neither for an explicit description of the corruptions. We refer the interested reader to [CCL15] for details.

Oblivious Transfer The ideal functionality of an Oblivious Transfer (OT) protocol was given in [Can01,CKWZ13,ABB$^+$13]. We recall it in simple UC in Figure 4 using the functionality introduced in [BCG16]. Note that the BPR model [BPR00] given for PAKE protocols can be adapted to give a game-based security model for OT schemes but this is well beyond the scope of this paper.

The party P_i is the sender $\mathcal{S}$, while the party P_j is the receiver $\mathbb{R}$. The former is provided with a database consisting of a set of n lines $(L_1, \ldots, L_n)$, while the latter is querying a particular line L_s (with $s \in \{1, \ldots, n\}$). Since there is no communication between them (the functionality deals with everything), it automatically ensures the oblivious property on both sides (the sender does not learn which line was queried, while the receiver does not learn any line other than L_s).

B Generic Construction For SPHF Friendly Commitment

We here give the generic SPHF-friendly commitment scheme from [ABB$^+$13] and [BC15], which is useful to obtain our concrete construction in Section 4.

B.1 Generic Construction

As before, $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ denotes an IND-CCA2 encryption scheme and $(\mathsf{KeyGen}, \mathsf{VKeyGen}, \mathsf{CH}, \mathsf{Coll}, \mathsf{Valid})$ a chameleon hash scheme. It should be feasible to compute a CCA-encryption of the opening value of the chameleon hash. We also

The functionality $\mathcal{F}_{(1,N)\text{-OT}}$ is parametrized by a security parameter κ. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_N$ via the following queries:

- **Upon receiving an input $(\texttt{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, (L_1, \ldots, L_N))$ from party P_i**, with $L_i \in \{0,1\}^\kappa$ for all i: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (L_1, \ldots, L_N))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further `Send`-message with the same ssid from P_i.
- **Upon receiving an input $(\texttt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ from party P_j**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (L_1, \ldots, L_N))$ is not recorded. Otherwise, reveal $(\texttt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$ and send $(\texttt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, L_s)$ to P_j and ignore further `Receive`-message with the same ssid from P_j.

Fig. 4. Ideal Functionality for 1-out-of-N Oblivious Transfer $\mathcal{F}_{(1,N)\text{-OT}}$

require the encryption to accept an SPHF on the language of valid ciphertexts, and the chameleon-hash to be verifiable by the receiver. This requires a pre-flow, in which the server is assumed to execute the algorithm VKeyGen to generate a verification key and its trapdoor and send the verification key to the sender[5]

Theorem 4 ([BC15]). *Given a verifiable collision-resistant chameleon hash and a secure* CCA*-encryption accepting an* SPHF *on the language of valid ciphertexts, the construction below provides a commitment scheme which is* SPHF*-friendly.*

- **Setup and simulated setup algorithms:** $\mathsf{SetupComT}(1^\kappa)$ (the algorithm for setup with trapdoors) generates the various parameters param, for the setting of the SPHF-friendly labelled CCA-encryption scheme and the chameleon hash scheme. It then generates the corresponding keys and trapdoors: $(\mathsf{ck}, \mathsf{tk})$ for the chameleon hash scheme and $(\mathsf{ek}, \mathsf{dk})$ for the encryption scheme.
  For $\mathsf{SetupCom}(1^\kappa)$ (the algorithm for setup without trapdoors), the setting and the keys are generated the same way, but forgetting the way the keys were constructed (such as the scalars, in a DDH-based setting), thus without any trapdoor.
  The algorithms both output the CRS $\rho = (\mathsf{ek}, \mathsf{ck}, \mathsf{param})$. In the first case, τ denotes the trapdoors $(\mathsf{dk}, \mathsf{tk})$.
- **Pre-flow (verification key generation algorithm):** player Q executes $\mathsf{VKeyGen}(\mathsf{ck})$ to generate the chameleon designated verification key vk and the trapdoor vtk and sends vk to the sender P.
- **Targeted commitment algorithm:** $\mathsf{Com}^\ell(\mathbf{M}; Q)$ from player P to player Q, for $\mathbf{M} = (M_i)_i \in \{0,1\}^t$ and a label ℓ, works as follows:
  - For $i \in [\![1, t]\!]$, it chooses r_{i,M_i} at random and computes $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, M_i; r_{i,M_i})$ to obtain the hash value a_i and the corresponding opening value d_{i,M_i}. It samples at random the values $r_{i,1-M_i}$ and $d_{i,1-M_i}$. We denote as $\mathbf{a} = (a_1, \ldots, a_m)$ the tuple of commitments and $\mathbf{d} = (d_{i,j})_{i,j}$.

[5] This makes the commitment not completely non-interactive, but this pre-flow will be merged with the first flow of our OT.

  - For $i \in [\![1, t]\!]$ and $j = 0, 1$, it gets $\mathbf{b} = (b_{i,j})_{i,j} = \mathsf{Encrypt}^{\ell'}_{\mathsf{ek}}(\mathbf{d}; \mathbf{s})$, where $\mathbf{s}$ is taken at random and $\ell' = (\ell, \mathbf{a})$.

  The commitment is $C = (\mathbf{a}, \mathbf{b})$, and the opening information is the m-tuple $\delta = (s_{1,M_1}, \ldots, s_{t,M_t})$.

– **Verification algorithm:** $\mathsf{VerCom}^{\ell}(\mathsf{vtk}, C, \mathbf{M}, \delta)$ first checks the validity of the ciphertexts b_{i,M_i} with randomness s_{i,M_i}, then extracts d_{i,M_i} from b_{i,M_i} and s_{i,M_i}, and finally checks the chameleon hash a_i with opening value d_{i,M_i}, for $i \in [\![1, t]\!]$, via the algorithm $\mathsf{Valid}(\mathsf{ck}, \mathsf{vk}, M_i, a_i, d_{i,M_i}, \mathsf{vtk})$.

– **Simulated targeted commitment algorithm:** $\mathsf{SimCom}^{\ell}(\tau; Q)$ from the simulator to player Q, takes as input the equivocation trapdoor, namely tk, from $\tau = (\mathsf{dk}, \mathsf{tk})$, and outputs the commitment $C = (\mathbf{a}, \mathbf{b})$ and equivocation key $\mathsf{eqk} = \mathbf{s}$, where

  - For $i \in [\![1, t]\!]$, it chooses $r_{i,0}$ at random, computes $(a_i, d_{i,0}) = \mathsf{CH}(\mathsf{ck}, \mathsf{vk}, 0; r_{i,0})$, and uses the equivocation trapdoor tk to compute $r_{i,1}$ used to open the chameleon hash to 1 such that $\mathsf{CH}(\mathsf{ck}, \mathsf{vk}, 1; r_{i,1})$ is equal to $(a_i, d_{i,1})$. This leads to $\mathbf{a}$ and $\mathbf{d}$, making $d_{i,j}$ the opening value for $a_{i,j}$ for all $i \in [\![1, t]\!]$ and $j = 0, 1$.
  - $\mathbf{b}$ is built as above: $\mathbf{b} = (b_{i,j})_{i,j} = \mathsf{Encrypt}^{\ell'}_{\mathsf{ek}}(\mathbf{d}; \mathbf{s})$, where $\mathsf{eqk} = \mathbf{s}$ is taken at random and $\ell' = (\ell, \mathbf{a})$.

– **Equivocation algorithm:** $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, \mathbf{M})$ simply uses part of the equivocation key eqk (computed by the SimCom algorithm) to obtain the opening information $\delta = (s_{1,M_1}, \ldots, s_{t,M_t})$ in order to open to $\mathbf{M} = (M_i)_i$.

– **Extraction algorithm:** $\mathsf{ExtCom}^{\ell}(\tau, \mathsf{vtk}, C)$ takes as input the extraction trapdoor, namely the decryption key dk, from $\tau = (\mathsf{dk}, \mathsf{tk})$, the verification trapdoor vtk and a commitment $C = (\mathbf{a}, \mathbf{b})$. For $i \in [\![1, t]\!]$ and $j = 0, 1$, it first extracts the value $d_{i,j}$ from the ciphertext $b_{i,j}$, using the decryption key dk. Then, for $i \in [\![1, t]\!]$, it checks the chameleon hash a_i with opening values $d_{i,0}$ and $d_{i,1}$ with the help of the algorithm $\mathsf{Valid}(\mathsf{ck}, \mathsf{vk}, j, a_i, d_{i,j}, \mathsf{vtk})$ for $j = 0, 1$. If only one opening value $d_{i,j}$ satisfies the verification equality of the chameleon hash, then $j = M_i$. If this condition holds for each $i \in [\![1, t]\!]$, then the extraction algorithm outputs $(M_i)_i$. Otherwise (either if $\mathbf{b}$ could not be correctly decrypted, or there was an ambiguity while checking $\mathbf{a}$, with at least one chameleon hash a_i with two possible opening values $d_{i,0}$ and $d_{i,1}$), it outputs $\perp$.

B.2 Building Blocks based on LWE

We now explain which building blocks are necessary to implement such an SPHF-friendly commitment scheme based on LWE, giving the explicit construction described in Section 4.

Chameleon Hash One simply uses the scheme described in Section 3. Recall that this scheme is based on SIS assumption. Since LWE is harder than SIS (see Section 2), this scheme is secure under LWE assumption as well.

$2t$-labelled multi LWE-based Encryption Scheme One starts by using the TAG-IND-CCA2 encryption scheme described in Section 3, which is secure under the hardness of the LWE assumption. One then uses the technique described in Section A.5. The strongly unforgeable one-time signature scheme needed will simply be instanciated by the above chameleon hash, using the technique described in Section 3.

Smooth Projective Hash Function Finally, the SPHF needed for each bit of the encryption will be obtained by first considering the approximated bit SPHF described in Section 3 and amplifying it as described in Section A.2 in order to obtain an approximate SPHF.

C Implementation

C.1 Parameter Choices

In our construction, we select the parameters q and n in order to achieve the desired level of security for the LWE-based schemes. In particular, the modulus q is chosen to be an odd prime. We take advantage of Albrecht's estimator[6] [APS15] which, at present, covers the following attacks: meet-in-the-middle exhaustive search, coded-BKW [GJS15], dual-lattice attack and small/sparse secret variant [Alb17], lattice reduction with enumeration [LP11], primal attack via uSVP [AFG14,BG14], Arora-Ge algorithm [AG11] using Gröbner bases [ACFP14].

Once q and n are chosen, we instantiate the remaining parameters using the computational instantiation of the trapdoor with $m = n(k+2)$.

- We take a "randomized-rounding parameter" $r \approx \sqrt{\ln(2/\epsilon)/\pi}$, where ϵ is a desired bound on the statistical error introduced by each randomized-rounding operation for $\mathbb{Z}$. Concretely, we use a parameter of $r = 4.5$ for $\mathbb{Z}$, which corresponds to statistical error of less than 2^{-90} for each operation [MP12].
- Certain Gaussian parameters are supported by reductions from worst-case lattice problems to LWE. The Gaussian parameter σ for the trapdoor sampling was originally stated as $\sigma = 2\sqrt{n}$ to ensure that the LWE instance of parameters n, q and σ is hard. In addition, the correctness of the TAG-IND-CCA2 encryption scheme is guaranteed on condition that $\sigma m^{3/2} \cdot r \leq q$. However, for efficiency reasons we follow the methodology as in [NAB$^+$17] to estimate the security level of our proposed parameters. In particular, if the Gaussian parameter αq of the LWE error sufficiently exceeds $\sqrt{\ln(N)/(2\pi)}$ where N is the number of discrete Gaussian samples, then the corresponding LWE problem is plausibly hard. Concretely, the threshold for Gaussian parameters αq corresponding to an extremely large bound $N \leq 2^{256}$ is ≈ 5.314.
- We have two different Gaussian parameters in the Chameleon Hash scheme, σ for the $\mathsf{TrapGen}$'s distribution and s for the randomness space's distribution. By [CHKP10], we use $s = \sigma r$.

[6] `https://bitbucket.org/malb/lwe-estimator`

PARAMETERS SET. We combine all the conditions to obtain the following set of parameters, used in our implementation. We ran the LWE security estimator [APS15] to find the lowest security levels for the uSVP, decoding, and dual attacks and selected the least value of the number of security bits κ for all 3 attacks on classical/quantum computers based on the estimates for the BKZ (quantum) sieve reduction cost model. Some possible choices of parameters are reported in Table 1.

Table 1. Security estimates for different choices of parameter sets (δ is the Root Hermite factor).

security level	κ		q	n	m	σ	δ
	quantum	classical					
medium	128	138	2147494753	976	33184	41	1.004226
	129	138	2147493889	1024	34816	25.5	1.004234
high	192	209	2147493889	1536	52224	21	1.002954

C.2 Implementation

Lattice-based constructions are known to be highly parallelizable. Therefore, we build our library so that it could be simultaneously called from concurrent threads. To this end, we make use of NTL's built-in thread pool. All floating-point computations are performed using double-precision arithmetic, as in [GM18].

The main bottleneck in the implementations is the Gaussian pre-image sampling operation, which is used the One-Time Signatures construction. Particularly, the sampling algorithm SampleD of [MP12] consists of two stages:

- an off-line stage, which generates perturbation vectors with covariance matrix defined by the trapdoor transformation $\mathbf{T}$,
- an on-line stage which generates Gaussian samples from a primitive (easy to sample) lattice $\mathbf{G}^n$.

We leverage the recent techniques described in [GM18,BFRLS18] to improve the complexity of the on-line stage, which is far more critical in applications. The dimension of the lattice was chosen $n = 976$, corresponding to the "medium" security level displayed in Table 1.

Our implementation was carried out in plain C++11, using the library NTL [Sho01] version 11.3.1 and the library GMP version 6.1.0 [Gt18] for handling generic number theory, and it is available at `https://github.com/vqhuy/lwe-ot`.