Errorless Smooth Projective Hash Function based on LWE

From worst-case Assumptions up to Universal Composability

Olivier Blazy[1] Céline Chevalier[2] Léo Ducas[3] Jiaxin Pan[1]

[1] Faculty of Mathematics
Horst Görtz Institute for IT-Security
Ruhr University Bochum, Germany
{olivier.blazy,jiaxin.pan}@rub.de
[2] Faculty of Mathematics, Universite Paris II
celine.chevalier@ens.fr
[3] Computer Science and Engineering, University of California San Diego
lducas@eng.ucsd.edu

Abstract

Smooth Projective Hash Functions are one of the base tools to build interactive protocols; and this notion has lead to the construction of numerous protocols enjoying strong security notions, such as the security in the Bellare-Pointcheval-Rogaway (BPR) model or even *Universal Composability* (UC).

Yet, the construction of SPHF has been almost limited to discrete-logarithm or pairing type assumptions up to now. This stands in contrast with domains such as homomorphic encryption or functional encryption, where *Lattice Based Cryptography* has already caught up and overtook discrete-log/pairing based cryptography. So far, work in the direction of UC based on lattices is almost restricted to a paper from Peikert, Vaikuntanathan, and Waters (Crypto 2008) dealing with Oblivious Transfer in the UC framework, and work in the direction of password-authenticated key exchange protocols (PAKE) to one from Katz and Vaikuntanathan (Asiacrypt 2009) on a 3-round Password-Authenticated Key Exchange, but restraining itself to the BPR model. It seems that dealing with errors in those contexts is not as easy as it is for encryption.

In this work, we identify the problem at its source, namely, the lattice version of Diffie-Hellman key exchange protocol: the key agreement is only approximate. We explicit a simple folklore trick to obtain true, *errorless*, one-round key exchange from LWE. We then show that this trick can be adapted to various lattice encryption schemes, leading, with some technicalities, to errorless SPHF's. From there, we derive three new results, namely the first lattice-based following protocols: a one-round PAKE secure in the BPR model, a 3-round PAKE secure in the UC model, and a UC commitment scheme, all of them based on SIS and LWE assumptions.

Keywords: LWE encryption, Lattices, Universal Composability, Password-Authenticated Key-Exchange, UC Commitment, Smooth Projective Hash Functions

1 Introduction

In the recent years, lattice based cryptography has been able to mimic many constructions from the discrete-logarithm/pairing world, and even go beyond. The incentives to switch to lattices are numerous; they offer strong security guarantees, such as hardness under worst-case assumptions, and potential resistance to quantum algorithms. They also offer better asymptotical security and good parallelism. And even more interestingly, lattice problems such as LWE have shown to be extremely versatile, allowing the construction of primitives which existence has been a long term open problem, such as Fully Homomorphic Encryption, following the breakthrough work of Gentry [29]. We also mention the recent proposal of Multi-Linear-Maps based on a new assumption related to lattice problems [26]. Also in the domain of functional encryption, LWE-based constructions have caught-up on, and then improved upon pairing-based cryptography, with the recent results of functional encryption for arbitrary circuits [31, 27].

Yet, there is one primitive, called Smooth Projective Hash Function, for which LWE based constructions are still far behind discrete-logarithm constructions. Smooth Projective Hash Functions (or SPHF) were introduced by Cramer and Shoup [21] in order to achieve IND-CCA security from IND-CPA encryption schemes, and generalised by Gennaro and Lindell [28]. They have recently shown to be quite useful for many other purposes, for example they allow implicit designated-verifier proofs of membership [2, 11], that is proof that a value verifies some relation, that can only convince one chosen verifier.

A simple application of SPHF —but extremely useful example in practice— is *Password-Authenticated Key Exchange* (it was proposed in the works of [35, 28], as part of what is now known as the KOY-GL paradigm). A PAKE is a protocol similar to key exchange, with the feature that it is successful only if the parties share a common password (which they usually commit to). This mechanism can offer security against offline dictionary attacks (formally called security in the Bellare-Pointcheval-Rogoway setting [7]), and therefore protecting users from being impersonated, even if they use low-entropy passwords.

Those PAKE protocols have recently reached the ultimate security notion called *Universal Composability* [18, 2, 37, 8]. The UC framework, introduced by Canetti [15] is a increasingly popular simulation-based security paradigm (but only studied by Peikert, Vaikuntanathan, and Waters [43] in the context of lattice-based protocols —to construct Oblivious Transfer). It guarantees that a protocol, possibly running concurrently with arbitrary —even insecure— protocols, proven secure in this framework remains secure. Such framework allows to split the design of a complex protocol into that of simpler sub-protocols. It is particularly useful in the case of PAKE, since it ensures that the protocol remains secure even in case a user mistypes or reuses the password (there is no particular assumption on the distribution of passwords).

PAKE protocols using the KOY-GL paradigm rely on a central cryptographic tool, called commitments. These commitments allow, in a two-phase protocol between two parties, a committer to give the receiver an *in silico* analogue of a sealed envelope containing a value m, and then the committer to reveal m so that the receiver can verify it. The security definition for commitment schemes in the UC framework was presented by Canetti and Fischlin [16]. Several UC-secure commitment schemes in the CRS model have been proposed since. Ostrovsky, and Sahai [19] proposed inefficient non-interactive schemes from general primitives. On the other hand, Damgård and Nielsen [22], and Camenisch and Shoup [14] (among others) presented interactive constructions from several number-theoretic assumptions.

Lindell [39] has recently presented the first very efficient commitment scheme proven in the UC framework. They can be viewed as combinations of Cramer-Shoup encryption schemes and Σ-protocols. This scheme has recently been revised by Blazy *et al.* in [10] where they reduce the number of rounds of the adaptive version. Fischlin, Libert and Manulis [25] adapted the scheme secure against static corruptions by using non-interactive Groth-Sahai proofs [32] to make it non-interactive. SPHF is one of the essential ingredients to construct those commitments with very strong properties [2, 1] in particular when in the context of PAKE schemes.

But as we mentioned earlier, building SPHF's from lattice assumptions has remained a open problem until now. Precisely none of the aforementioned constructions are based on worst-case assumption such as the lattice problems LWE or SIS. The only exception we know of are works of Katz and Vaikuntanathan [36], later improved by Ding and Fan [23]. It is the first known PAKE in the standard model based on lattices; but it requires 3-rounds an can only be proven secure in the BPR [7] setting. The authors proceed as follows: first, they construct an *approximate* SPHF, and from there, derive their PAKE. The fact that the SPHF is only approximate seems to be at least painful technicality, if not a real issue to construct stronger, or better schemes.

1.1 This work

In this work we show how to construct an *errorless* (as opposed to approximate) SPHF from lattice based assumption (namely LWE and SIS), and derive several applications out of it. To be precise, we call our construction errorless in the sense that the result is exact most of the time (i.e. except with negligible probability), as opposed to *approximate* construction, that always contains a small error.

SPHF construction, Main idea To build such an SPHF, we first analyse where the issue of approximation arise between the simplest protocols from both the discrete-log world and lattice world. At high level, the so called Regev's Dual Encryption scheme, can be seen as a lattice analogue of ElGamal. Changing one's point of view, allows one to see ElGamal encryption as key Exchange protocol, precisely, a Diffie-Hellman one-round Key Exchange. If one applies the same change of viewpoint on Regev's Dual scheme, one only obtains an *approximate* key exchange protocol; and to be useful, the parties need an extra round to get rid of the error. In some more advanced protocols, this error might simply be impossible to handle.

Still, one notices the fact that as an encryption scheme, the Dual Regev cryptosystem is not *approximate*, and decryption almost always leads to the original ciphertext. This is due to the use of a simple error correcting code ECC; first step of decryption leads to a noisy codeword; and by design one can recover from this error: $\mathsf{ECC}^{-1}(\mathsf{ECC}(M) + e) = M$ for any message M and small error $e \in \mathbb{Z}_q$. Yet, to get exact key exchange from this $\mathsf{ECC} : \{0,1\} \to \mathbb{Z}_q$ one would require that $\mathsf{ECC}^{-1}(x + e) = \mathsf{ECC}^{-1}(x)$ for any x and any small errors e. Obviously this would implies that ECC^{-1} is a constant function. A folklore workaround is to restrict this requirement to hold only for all but a negligible fractions of the x's by tolerating negligible probability of failure of the protocol. This becomes actually doable when the set of the x's is superpolynomially larger that the set of errors.

SPHF construction, Additional Technicalities While we warm with building an errorless SPHF on the CPA-secure Dual Regev Encryption scheme, the real goal for applications is to build and errorless SPHF over a lattice based CCA scheme (or equivalently, a Tag-Based Encryption scheme). Our construction is based on the trapdoored scheme of Micciancio and Peikert, yet we need to implement a modifications: to be compatible with SPHF construction, one rather encodes the message in the higher bits of the ciphertext rather than in the lower bits.

PAKE and LAKE Once those SPHF are created, our new results comes almost generically following techniques introduced in [8, 9]. One just have to supersede the CCA-1 encryption with a Strong-One Time Signature to achieve CCA-2 security, and then adapt the known technique Double Cramer-Shoup [8]. Yet, unlike the analog construction from pairings, messages are only made of one bit. To compensate the small message space size, we send a linear number of challenges and parties have to solve all of them at once. This allows us to achieve the first One-Round PAKE on Lattices, and the first PAKE

UC protocol on Lattices (with only 3 rounds). Both PAKE can even be extended to *Language-Authenticated-Key-Exchange* for a certain class of languages.

UC commitment Once we have our Double Micciancio-Peikert like encryption —our lattice Analogue of the Double Cramer-Shoup— with just an additional twist we can manage to create a UC Commitment by simply adapting Lindell's commitment to provide the first Lattice-Based UC-Commitment, furthermore in the adaptive setting.

Closing remarks It is unclear if it is really necessary to get rid of approximation and resort to an errorless SPHF to obtain such protocols based on lattice problems with security in the UC model. Yet, our new SPHF certainly is a convenient tool, offering a simpler and more modular interface for the construction, the security proofs, and the understanding of such advanced protocols. It remains that parts of the proofs needs to be redone, because we lack truly generic result relating SPHF, PAKE, LAKE and commitments; we believe this would be an interesting open problem.

1.2 Outline of the paper

The construction of the paper is straightforward, we first start in Section 2, by recalling some useful definitions, results and constructions, then in Section 3, we construct Smooth Projective Hash Functions, first on the Dual Regev Encryption scheme, and then on an ad-hoc version of the Micciancio-Peikert. In Section 4, we then combine this encryption and the associated SPHF to create two PAKE, first in 3 rounds in the UC framework, then in 1 round in the BPR model, finally in Section 5, we also use this encryption to provide the first Lattice-based UC commitment.

Appendices follow a similar order, we first start with additional definitions, results and constructions in the Appendix A, then present a doubled version of our version of the Micciancio-Peikert encryption in the Appendix B. We then proceed to introduce the UC framework and the required functionalities in the Appendix C, and give various proofs in the Appendix D. A Cheat Sheet is given on the last page in the Appendix E to give a clear parallel between classical instantiations on pairing, on the one we use here on Lattices.

2 Technical preliminaries

In this section, we define some useful tools for our constructions in the following sections. These useful tools include Approximate Key Exchange, Tag-based Encryption, Trapdoor Commitment and Strong One-Time Signature. All the schemes here are constructed from Lattice Assumptions. Due to lack of space, we recall the security notions and the corresponding Lattice Assumptions in Appendix A.

2.1 Lattice Problems

Definition 2.1 (The Short Integer Solution Problem, SIS) *The Short Integer Solution problem* $\mathsf{SIS}_{n,m,q,\beta}$*, with* m *unknowns,* $n \leq m$ *equations modulo* q *and norm-bound* β *is as follows: given a random matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *chosen uniformly, find a non-zero short vector* $\mathbf{v} \in \mathbb{Z}_q^m \setminus \{\mathbf{0}\}$ *such that* $\mathbf{A}\cdot\mathbf{v} = \mathbf{0}$ *and* $\|\mathbf{v}\| \leq \beta$*.*

Definition 2.2 (The Learning with Errors Problem, dLWE, decisional version) *The Learning with Errors Problem, decisional version,* $\mathsf{dLWE}_{n,m,q,\chi}$*, with* n *unknown,* $m \geq n$ *samples, modulo* q *and with errors distribution* χ *is as follows: for a random secret* $\mathbf{s}$ *uniformly chosen in* $\mathbb{Z}_q^n$*, and given* m *samples either all of the form* $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a}\rangle + e)$ *where* $e \leftarrow \chi$*, or from the uniform distribution* $(\mathbf{a}, b) \leftarrow \mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)$*; decide if the samples comes from the former or the latter case.*

Worst-case to Average-Case Connection Both SIS and LWE enjoy strong security guarantees for certain range of parameters. Precisely, SIS is as hard in the average-case as certain lattice problems (unique-SVP or SIVP) in the worst-case, as proved originally by Ajtai and later generalized [3, 42, 30]. On the other hand the problem LWE was popularized by the work of Regev [44], proving similar average-case to worst-case reduction from LWE to gap-SVP, but the reduction is a quantum algorithm. Of course, the parameters of the underlying lattice problems depends on the parameters of SIS and LWE; see [44, 42] for the precise statement.

Error Distribution The hardness results of [44] for LWE hold for Discrete Gaussian error distributions $\chi = D_{\mathbb{Z},s}$, where s denotes (up to a constant factor) the standard deviation. Due to space restriction, we defer definition to appendix, definition A.8. For our purpose, we only require a bound on the size of vectors sampled according to that distribution:

Lemma 2.3 (Tailcut of discrete Gaussians [5, 42]) *For any* $s > 0$ *and* $c > 1/\sqrt{2\pi}$*, and any* m*-dimensional lattice* L*, and vector* $\mathbf{x} \in \mathbb{R}^n$*,* $\rho_s((L+\mathbf{x}) \setminus c\cdot s\sqrt{m}\mathfrak{B}) < 2C^m\rho_s(L)$*, where* $C = c\sqrt{2\pi e}\cdot e^{-\pi c^2} < 1$*, and* $\mathfrak{B}$ *is the centered unit ball. In particular, for* $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,s}$*,* $\|\mathbf{x}\| \leq s\sqrt{m}$ *except with negligible probability in* m*.*

Strong LWE. In typical LWE based encryption, the parameters are chosen as follow: $n = \Omega(\lambda)$, $q \leq 2^{\mathrm{poly}(n)}$, $m = O(n \log q)$ and the error distribution to be the discrete Gaussian $D_{\mathbb{Z},s}$ of standard deviation $s = q/\mathrm{poly}(n)$. For such parameters, the best known attacks runs in time $2^{\Theta(\lambda)}$. Yet, one might use smaller error parameter s, at the cost of some security; this was necessary for the first constructions of FHE from LWE [12] (but was recently overcome [13]). It was also the case for the early result of Banerjee *et al.* [6] to prove the hardness of *Learning with Rounding*[1]. Actually, the reason we need such parameters is similar to the one of [6]: we wish that for a uniform random value $x \in \mathbb{Z}_q$, adding an error $e \leftarrow \mathbb{Z}, s$ to x doesn't modifies its higher bit, except maybe with negligible probability.

Specifically, polynomial attacks are known when s is as small as $q/2^{\Theta(n)}$, yet if $s \geq q/f(n)$ for a sub-exponential function the best known attacks are subexponential. For the sake of clarity, one could set our parameters to $q = 2^{\Theta(n)}$, and $s = q/2^{\Theta(\sqrt{n})}$; for all we know, those parameters may well require $2^{\Theta(\sqrt{n})}$ time to break; and it will guarantee that our protocols fails with negligible probability $2^{-\Theta(\sqrt{n})}$; that is to obtain λ bits of security on should choose $n = \Theta(\lambda^2)$, which does impact the efficiency in a significant way. It is indeed convenient to think of protocols with negligible probability of failure in theory, especially in the UC model: it allows one to repeat sequentially or concurrently polynomially many instances of various protocols, while maintaining overall failure probability negligible. In practice, the situation might not be so catastrophic: in many scenario an error probability of 2^{-30} is perfectly acceptable for a 128-bit secure cryptosystem.

2.2 The Crux of Matter: Achieving Errorless Key Exchange

We now discuss the question of approximation in relation to LWE based Key exchange. While the passively secure Key Exchange protocol presented below is not formally used for our construction, it offers a simple context to give intuition on problem related to error and our proposed solution.

LWE Encryption and Approximate Key Exchange Before giving formal definitions, let us re-interpret the LWE encryption scheme of Regev [44] as a "approximate key exchange". Let n denotes the security parameter, and let $q \leq 2^{\mathrm{poly}(n)}$ and $m = \mathrm{poly}(n)$ be such that $m \geq O(n \log(q))$. Assuming the parties do have a common reference string parsed as a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, it is possible to re-interpret the previous scheme as a an approximate 1-round key exchange protocol as explained in Figure 1.

Alice		**Bob**
$\mathbf{t} \leftarrow \mathcal{U}(\{0,1\}^m)$		$\mathbf{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n), \mathbf{e} \leftarrow D_{\mathbb{Z},s}^m$
Compute $\mathbf{a} = \mathbf{A} \cdot \mathbf{t} \in \mathbb{Z}^n$		Calcul $\mathbf{b} = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} \in \mathbb{Z}^m$
	$\xleftarrow{\mathbf{a}}$	
	$\xrightarrow{\mathbf{b}}$	
Compute $k_A = \langle \mathbf{t}, \mathbf{b} \rangle \in \mathbb{Z}_q$		Compute $k_B = \langle \mathbf{a}, \mathbf{s} \rangle \in \mathbb{Z}_q$

Figure 1: Approximate Key Exchange from LWE

At the end of this protocol, one notices that we have $k_A = k_B + \langle \mathbf{t}, \mathbf{e} \rangle$. Since $\mathbf{t}$ and $\mathbf{e}$ are small, we have $k_A \approx k_B$, yet for any third party that doesn't know anything about random values $\mathbf{t}, \mathbf{s}, \mathbf{e}$, k_A and k_B are computationally indistinguishable from random under the decisional LWE assumption.

There are now two ways to use this approximate exchanged key to have an exact shared secret.

LWE Encryption The typical encryption scheme (as in [44]) would use k_A as a one-pad together with a (very simple) error correcting code to build the cipher $c = \mathsf{ECC}(M) + k_A$; decryption procedure would use k_B to recover the message $M' = \mathsf{ECC}^{-1}(c - k_B) = \mathsf{ECC}^{-1}(\mathsf{ECC}(M) + \langle \mathbf{t}, \mathbf{e} \rangle)$. The error-correction for binary messages is simply defined as follow:

Definition 2.4 For any positive integer q, we define the error correcting code ECC and the correcting function ECC^{-1}

$$\mathsf{ECC}(M \in \{0,1\}) = \left\lfloor \frac{q}{2} \right\rfloor \cdot M \in \mathbb{Z}_q \quad \text{and} \quad \mathsf{ECC}^{-1}(x \in \mathbb{Z}_q) = \begin{cases} 0 & \text{if } x \in \{\lfloor -\frac{q}{4} \rfloor \ldots \lfloor \frac{q}{4} \rfloor\} \\ 1 & \text{otherwise} \end{cases}$$

One easily verifies that decryption is correct as soon as parameter choice are such that $|\langle \mathbf{t}, \mathbf{e} \rangle| < q/4$. In particular, assume the error is Gaussian of standard deviation s, then, except with probability negligible in n, one has $|\langle \mathbf{t}, \mathbf{e} \rangle| \leq \sqrt{nm}s$: correctness can be guaranteed for $s = q/\mathrm{poly}(n)$. Later on, we will focus on the dual version of that encryption scheme; that is after the key exchange of Figure 1, Bob encrypt $c = \mathsf{ECC}(M) + k_B$ and Alice Decrypt.

Both this encryption scheme and its dual naturally leads to an exact two-round Key exchange protocol: one party simply choose at random the shared key and send it encrypted to the other party.

[1] It was also partially overcome in [4], but only for bounded amount of LWR samples, which is not enough for the constructions of [6].

One Round Key Exchange from LWE with super-polynomially small error It is in fact possible to avoid adding a second round to obtain exact key exchange, by allowing failure to happen with negligible probability, and basing ourself on the strong version of LWE, that is when the error to modulus ratio s/q is polynomially small.

Notice that at the end of the KE protocol of fig. 1, $k_A = k_B + \langle \mathbf{t}, \mathbf{e} \rangle$, where $\mathbf{k}_B$ is uniformly random and $\langle \mathbf{t}, \mathbf{e} \rangle$ is small. The intuition is that, if we round both k_A and k_B, there is some probability that both rounded value will be equal. More precisely, using our previous error correcting code:

Lemma 2.5 *Let $q \in \mathbb{Z}$ and $\Delta \geq 1$ be implicit functions of λ such that q/Δ is superpolynomial in λ. Then, for any $e \in \mathbb{R}$ such that $|e| \leq \Delta$, with probability $1 - \mathsf{negl}(\lambda)$ over the choice of $x \leftarrow \mathcal{U}(\mathbb{Z}_q)$ one has*

$$\mathsf{ECC}^{-1}(x+e) = \mathsf{ECC}^{-1}(x) \quad \text{and} \quad \mathsf{ECC}^{-1}\left(x \pm \lfloor \frac{q}{2} \rfloor + e\right) - \mathsf{ECC}^{-1}(x) = 1 \bmod 2$$

Proof: Notice that, $x \notin \{-\Delta, \ldots, \Delta\} \cup \{\lfloor \frac{q}{2} \rfloor - \Delta, \ldots, \lfloor \frac{q}{2} \rfloor + \Delta\} \Rightarrow \mathsf{ECC}^{-1}(x) = \mathsf{ECC}^{-1}(x+e)$. We conclude that $\mathsf{ECC}^{-1}(x+e) = \mathsf{ECC}^{-1}(x)$ holds except with probability $4\Delta/q = \mathsf{negl}(\lambda)$. The second equation is similar. ∎

Since k_A is uniformly random, we have $\mathsf{ECC}^{-1}(k_A) = \mathsf{ECC}^{-1}(k_B)$ except with negligible probability $4\Delta/q = \mathsf{negl}(\lambda)$; in other words, we can obtain exact key exchange that fails only with negligible probability as shown in Figure 2.

Alice		**Bob**
$\mathbf{t} \leftarrow \mathcal{U}(\{0,1\}^m)$		$\mathbf{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n), \mathbf{e} \leftarrow D_{\mathbb{Z},s}^m$
Compute $\mathbf{a} = \mathbf{A} \cdot \mathbf{t} \in \mathbb{Z}^n$		Compute $\mathbf{b} = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} \in \mathbb{Z}^m$
	$\xleftarrow{\mathbf{a}}$	
	$\xrightarrow{\mathbf{b}}$	
Compute $k_A = \mathsf{ECC}^{-1}(\langle \mathbf{t}, \mathbf{b} \rangle) \in \mathbb{Z}_2$		Compute $k_B = \mathsf{ECC}^{-1}(\langle \mathbf{a}, \mathbf{s} \rangle) \in \mathbb{Z}_2$

Figure 2: Exact Key Exchange from LWE

The previous idea is highly related to the work of [6] especially their hardness proof of LWR; yet we were unable to build a better key exchange directly from LWR. Indeed, even if, in Figure 1, one replaces Bob's random errors e by a deterministic rounding error, the protocol still doesn't converge exactly.

2.3 Trapdoor Commitment / One Time Signature

In the Appendix A.2.1, we detail the definition and construction of a Trapdoor Commitment, and we explain that such a commitment is also a Strong One-Time Signature.

As explained in [38, 17], a tag-based encryption scheme can be transferred into a CCA-2 public key encryption by using a Strong One-time Signature following the construction in Figure 3.

Lemma 2.6 *Assuming the TBE scheme is selective-tag chosen-ciphertext secure, the OTS is a strong, one-time signature scheme, then the public-key encryption scheme presented in Figure 3 is chosen-ciphertext secure.*

3 Errorless Smooth Projective Hash Functions from LWE

Smooth Projective Hash functions were initially introduced by Cramer and Shoup [21]. Since then, they have come in different flavors, up to the [37]-Smooth Projective Hash Functions, and an explicit instantiation in the case of Cramer-Shoup Ciphertext was given in [8].

In this section, we present a new instantiation of such Smooth Projective Hash Functions on Lattices. These SPHF being based on Lattices encryption around LWE, they inherit the same drawbacks, and so are limited by the existence of possibly invalid decommitments, which leads to an approximate correctness. With negligible probability the correctness will not be fulfilled, but with overwhelming probability the values computed are exactly the same when users are honest.

For the rest of the section, we let ECC be the error correcting code as in Definition 2.4. The parameters will be assumed to be as follows: $n = \mathrm{poly}(\lambda)$, $q = 2^{\Theta(n)}$, $m \geq \Theta(nk)$, and $s = q/f(n)$ where f is superpolynomial but subexponential in λ, in particular $s = q \cdot \mathsf{negl}(\lambda)$.

$\mathsf{Encrypt}(\mathsf{ek}, M)$: Generates: $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{OTS.Gen}(1^n)$ Computes: $C \leftarrow \mathsf{TBE.Encrypt}(\mathsf{ek}, \mathsf{vk}, M)$ $\sigma \leftarrow \mathsf{OTS.Sign}(\mathsf{sk}, C)$ Returns $\mathcal{C} = (C, \mathsf{vk}, \sigma)$.	$\mathsf{Decrypt}(\mathsf{dk}, (C, \mathsf{vk}, \sigma))$: Check that $\mathsf{OTS.Verify}(\mathsf{vk}, C, \sigma)$ If verifies correctly, computes: $M = \mathsf{TBE.Decrypt}(\mathsf{dk}, \mathsf{vk}, C)$

Figure 3: Generic Construction of a CCA-2 encryption from a TBE and a Strong-OTS.

3.1 SPHF On Dual Regev Encryption

To warm up, we start by presenting an Approximate Smooth Projective Hash Function on LWE encryption. A cheat-sheet detailing the parallel with SPHF on ElGamal is provided in App. E for readers used to discrete-log based protocols.

3.1.1 Dual Regev Encryption

- $\mathsf{KG}(1^n)$: Choose a uniform random matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times m}$ and $\bar{\mathbf{t}}$ uniform in $\{0,1\}^m$; set $\mathbf{a} = \bar{\mathbf{A}} \cdot \bar{\mathbf{t}} \in \mathbb{Z}_q^n$, and $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{a}] \in \mathbb{Z}_q^{n\times(m+1)}$. Output the key pair $(\mathsf{dk} = \bar{\mathbf{t}}, \mathsf{ek} = \mathbf{A})$.
- $\mathsf{Encrypt}(M \in \{0,1\}, \mathsf{ek} = \mathbf{A} \in \mathbb{Z}_q^{n\times(m+1)}; \mathsf{wit} = (\mathbf{s},\mathbf{e}) \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \times D_{\mathbb{Z},s}^{m+1})$: output the ciphertext $\mathbf{c} = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} + [\mathbf{0}_m|\mathsf{ECC}(M)] \in \mathbb{Z}_q^{m+1}$, where $\mathbf{0}_m$ is the zero vector in $\mathbb{Z}_q^m$.
- $\mathsf{Decrypt}(\mathbf{c} \in \mathbb{Z}_q^{m+1}, \mathsf{dk} = \bar{\mathbf{t}} \in \{0,1\}^m)$: consider $\mathbf{t}^\top = [-\bar{\mathbf{t}}^\top|1]$, and output $M' = \mathsf{ECC}^{-1}(\langle \mathbf{t}', \mathbf{c}\rangle) \in \{0,1\}$

Correctness Let $(\mathsf{dk},\mathsf{ek}) \leftarrow \mathsf{KG}(1^n)$, $\mathbf{c} \leftarrow \mathsf{Encrypt}(M, \mathsf{ek} = \mathbf{A}; (\mathbf{s},\mathbf{e}) \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \times D_{\mathbb{Z},s}^{m+1})$ and finally $M' = \mathsf{Decrypt}(\mathbf{c}, \mathsf{dk} = \bar{\mathbf{t}})$. Note that, by construction of $\mathsf{ek} = \mathbf{A} \in \mathbb{Z}_q^{n\times(m+1)}$, that $\mathbf{t}^\top = [-\bar{\mathbf{t}}^\top|1]$ verifies $\mathbf{A}\cdot\mathbf{t} = \left[\bar{\mathbf{A}}|\bar{\mathbf{A}}\cdot\bar{\mathbf{t}}\right] \cdot \left[\begin{smallmatrix} -\bar{\mathbf{t}} \\ 1 \end{smallmatrix}\right] = \mathbf{0}_n$. Therefore

$$\begin{aligned} M' &= \mathsf{ECC}^{-1}(\langle \mathbf{t}, \mathbf{c}\rangle) = \mathsf{ECC}^{-1}\left(\mathbf{t}^\top \cdot (\mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} + [\mathbf{0}_m|\mathsf{ECC}(M)]\right) \\ &= \mathsf{ECC}^{-1}\left(\langle \mathbf{t}, \mathbf{e}\rangle + \mathsf{ECC}(M)\right) = M \end{aligned}$$

where the last equality follows from the fact that $\langle \mathbf{t}, \mathbf{e}\rangle \le s(m+1) \le q/4$ (Lemma 2.3 bounds $\|\mathbf{e}\| \le s\sqrt{m+1}$ and $\mathbf{t}$ is ternary).

CPA security (sketch). Use Leftover Hash Lemma to argue that $\mathbf{a}$ is uniform and independent of $\bar{\mathbf{A}}$: $\mathbf{A}$ is uniform, in particular it looks like a valid LWE matrix. Then the reduction to dLWE is straightforward.

3.1.2 A new SPHF on Dual Regev Encryption

We are now going to present a SPHF on the Dual Regev Encryption, while we are not going to use it directly in our constructions, this is a nice ground to the see the technical difficulties on building a Smooth Projective Hashing System on Lattices. For readers that are mostly familiar with discrete-log based protocols, a step by step comparison between how SPHF on ElGamal and SPHF on Dual Regev is provided in the Appendix E. In our upcoming applications, we are going to encrypt using an encryption key part of the crs. Users will be expected to do an implicit decommitment of their ciphertext. In other words, they are going to convince a verifier that their ciphertext is indeed a valid encryption of message M.

To do so, we are now going to build a Smooth Projective Hash Function on language of valid Dual Regev Encryptions of a message.

- HashKG: Output a uniformly chosen $\mathsf{hk} = \mathbf{v} \leftarrow D_{\mathbb{Z},\omega(\sqrt{\log\lambda})}^{m+1}$
- $\mathsf{ProjKG}\left(\mathsf{ek} = \mathbf{A} \in \mathbb{Z}^{n\times(m+1)}, \mathsf{hk} = \mathbf{v} \in D_{\mathbb{Z},\omega(\sqrt{\log\lambda})}^{m+1}\right)$: output $\mathsf{hp} = \mathbf{w} = \mathbf{A}\cdot\mathbf{v} \in \mathbb{Z}_q^n$
- $\mathsf{ProjHash}(\mathsf{hp} = \mathbf{w} \in \mathbb{Z}_q^n, \mathsf{wit} = \mathbf{s} \in \mathbb{Z}_q^n)$: output $H' = \mathsf{ECC}^{-1}(\langle \mathbf{w}, \mathbf{s}\rangle) \in \{0,1\}$
- $\mathsf{Hash}(\mathbf{c} \in \mathbb{Z}_q^{m+1}, M', \mathsf{hk} = \mathbf{v})$: output $H = \mathsf{ECC}^{-1}\left(\left\langle \mathbf{v}, (\mathbf{c} - [\mathbf{0}_m|\mathsf{ECC}(M')]^\top)\right\rangle\right) \in \{0,1\}$

For correctness and smoothness, we check that

$$\begin{aligned} H' - H &= \mathsf{ECC}^{-1}(\langle \mathbf{w}, \mathbf{s}\rangle) - \mathsf{ECC}^{-1}(\langle \mathbf{v}, \mathbf{c} - [\mathbf{0}_m|M]^\top\rangle) \\ &= \mathsf{ECC}^{-1}(\langle \mathbf{A}\cdot\mathbf{v}, \mathbf{s}\rangle) - \mathsf{ECC}^{-1}(\langle \mathbf{v}, \mathbf{A}^\top\cdot\mathbf{s} + \mathbf{e} + [\mathbf{0}_m|\mathsf{ECC}(M)] - [\mathbf{0}_m|\mathsf{ECC}(M')]\rangle). \end{aligned}$$

Correctness We consider the language of honest encryption of M, so we handle only encryption of M with a reasonably small randomness, by opposition to all ciphertexts that can be decrypted to M. Correctness is here to protect honest prover, so this restriction still keeps this property. (Invalid) Encryption of M with a randomness too big are part of the language but are only adversary generated, so their output can be wrong without impacting a normal execution. [2]

If $M = M'$, we have $H' - H = \mathsf{ECC}^{-1}(\langle \mathbf{A}\cdot\mathbf{v}, \mathbf{s}\rangle) - \mathsf{ECC}^{-1}(\langle \mathbf{A}\cdot\mathbf{v}, \mathbf{s}\rangle + \langle \mathbf{v}, \mathbf{e}\rangle)$. Notice that $\langle \mathbf{A}\cdot\mathbf{v}, \mathbf{s}\rangle$ follows a uniform distribution over $\mathbb{Z}_q$ the randomness of $\mathbf{s}$, while $\langle \mathbf{v}, \mathbf{e}\rangle \le \|\mathbf{v}\|\,\|\mathbf{e}\|$. Since $\mathbf{v}$ is binary, we have $\|\mathbf{v}\| \le \sqrt{m+1}$, while $\|\mathbf{e}\| \le 2\sqrt{m+1}\cdot s$ except with probability negligible in n, according to Lemma 2.3. We conclude using Lemma 2.5 that $H = H'$ except with probability negligible in n.

[2] For reference, [36] call this sub-language $\bar{L} \subset L$.

Smoothness We now have to handle two different cases:

- Valid encryption of $M \neq M'$, then $\mathsf{ECC}(M) - \mathsf{ECC}(M') = \pm\lfloor \frac{q}{2} \rfloor$; therefore

$$H - H' = \mathsf{ECC}^{-1}\left(\langle \mathbf{A} \cdot \mathbf{v}, \mathbf{s} \rangle\right) - \mathsf{ECC}^{-1}\left(\langle \mathbf{A} \cdot \mathbf{v}, \mathbf{s} \rangle \pm \lfloor \frac{q}{2} \rfloor v_{m+1} + \langle \mathbf{v}, \mathbf{e} \rangle\right) = v_{m+1} \bmod 2$$

  where the last equality holds except with negligible probability, following the same argumentation as above based on Lemmata 2.3 and 2.5. v_{m+1} is perfectly hidden under the leftover hash lemma, hence an adversary cannot compute a valid H' for a word outside a language with an advantage significantly bigger than a random guess.

- Invalid Encryption of $M' \neq M$, in this case this adversary knows that $\langle \mathbf{v}, \mathbf{e} \rangle$ may not be corrected automatically, however he now have to determine the value of $\bar{v}_{m+1}$, so the previous analysis still leads to the smoothness.

Pseudo-Randomness Under the CPA security of the encryption scheme, we can transform a commitment to a word in the language, to a commitment to a word outside of the language, hence an adversary against the pseudo-randomness either breaks the smoothness (which, information theoretically he cannot) or the indistinguishability of the encryption.

3.2 SPHF On Micciancio-Peikert CCA

In this section, we detail our construction of a SPHF over the CCA-1 (or equivalently a Tag-Based Encryption scheme) scheme of Micciancio and Peikert, with the following two key modifications to be compatible with our SPHF. First, for the correctness of the SPHF, we need to restrict the space of Tag-matrices $\mathcal{T}$ in comparison with the original construction[3]. Secondly, while the original scheme encodes the message in the lower-order bits of the ciphertext, we need it to be encoded in the higher order bits following the procedure described earlier.

For the rest of the section, we let $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ be the gadget matrix as defined in [41] and let ECC be the error correcting code as in Definition 2.4. The parameter will be assumed to be as follows: $n = \mathrm{poly}(\lambda)$ and it is a power of 2, $q = 2^k$ for some integer $k = \Theta(\lambda)$, $\bar{m} = \Theta(nk)$, $m = \bar{m} + nk$ and finally $s = q/f(\lambda)$ where f is superpolynomial but subexponential in λ, in particular $s = q \cdot \mathsf{negl}(\lambda)$.

3.2.1 Micciancio-Peikert CCA scheme as Tag-based Encryption

One of the keystone results of the work of Micciancio and Peikert [41] is the construction of a trapdoor for a tagged LWE function. It can be stated as follows. Considering space restriction, we limit ourself to this statement, the interested reader can refer to the article for a detailed construction. Once again, the reader used to discrete-log based SPHF might find our cheat-sheet in App. E quite helpful.

Theorem 3.1 (Reformulation of Theorem 4 from [41]) *With the parameters as above, there exists an implicit function $B = \mathrm{poly}(\lambda)$ and an algorithm* Invert*, that given inputs $\mathbf{R} \in \mathbb{Z}_q^{\bar{m} \times nk}$, $\mathbf{A} = [\bar{\mathbf{A}} | -\bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$, an invertible matrix $\mathbf{T} \in \mathbb{Z}_q^{n \times n}$ and a vector $\mathbf{b} = (\mathbf{A} + [\mathbf{0}|\mathbf{TG}])^\top \cdot \mathbf{s} + \mathbf{e} \in \mathbb{Z}^m$ for some $(\mathbf{s}, \mathbf{e}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^m$ outputs this couple $(\mathbf{s}, \mathbf{e})$ under the condition that $\|\mathbf{e}\| \leq q/B\sqrt{s_1(\mathbf{R})^2 + 1}$, where s_1 denotes the largest singular value of $\mathbf{R}$, except with negligible probability over the choice of $\bar{\mathbf{A}} \leftarrow \mathcal{U}(\mathbb{Z}_q^{\bar{m} \times n})$.*

Tag space Let $\mathcal{T} \subset \mathbb{Z}_q^{n \times n}$ denote the tag space. For the constructions of [41], the requirement on the set $\mathcal{T}$ are that, for all $\mathbf{T} \in \mathcal{T}$, $\mathbf{T}$ is invertible; and, for two distinct $\mathbf{T}, \mathbf{T}' \in \mathcal{T}$, $\mathbf{T} - \mathbf{T}'$ is invertible and finally that it is large: $|\mathcal{T}| \geq 2^{O(n)}$. A construction for such a tag-space is given in [41].

Tag Based Encryption The tag-based encryption scheme [41] is based on the trapdoor generation technique presented in Lemma A.14. The encryption $\mathsf{MP}^* = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\{0,1\}^{nk}$ is defined as follows:

- $\mathsf{KG}(1^n)$: choose a random matrix $\bar{\mathbf{A}} = \mathcal{U}(\mathbb{Z}_q^{n \times \bar{m}})$ and $\mathbf{R} \leftarrow D^{\bar{m} \times nk}_{\mathbb{Z}, \omega(\sqrt{\log \lambda})}$. Define the encryption key $\mathsf{ek} = \mathbf{A} = [\bar{\mathbf{A}} | -\bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$ and the decryption key $\mathsf{dk} = (\mathbf{R}, \mathbf{A})$.

- $\mathsf{Enc}(\mathsf{ek} = \mathbf{A}, \mathsf{tag} = \mathbf{T} \in \{0, \pm 1\}^{n \times n}, M \in \{0,1\}; \mathsf{wit} = (\mathbf{s}, \mathbf{e}) \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \times D_{\mathbb{Z}^m, s})$: Output

$$\mathbf{c} = (\mathbf{A} + [\mathbf{0}|\mathbf{TG}])^\top \cdot \mathbf{s} + \mathbf{e} + [\mathbf{0}|\mathsf{ECC}(M)]^\top \in \mathbb{Z}_q^m.$$

- $\mathsf{Dec}(\mathsf{dk} = (\mathbf{R}, \mathbf{A}), \mathsf{tag} = \mathbf{T}, \mathbf{c})$: Compute

[3] We mention that it is possible to build a SPHF without this requirement, but it would force one to know the Tag used for encryption at the projection phase ProjKG of the SPHF. While this is not a deal-breaker, it prevents us from achieving PAKE in one round, but would still be acceptable for the 3 round scheme.

1. $(\mathbf{s}_0, \mathbf{e}_0) = \mathsf{Invert}(\mathbf{R}, \mathbf{A}, \mathbf{T}, \mathbf{c})$ and $(\mathbf{s}_1, \mathbf{e}_1) = \mathsf{Invert}(\mathbf{R}, \mathbf{A}, \mathbf{T}, \mathbf{c} - [\mathbf{0}|\mathsf{ECC}(1)])$, so that

$$\mathbf{c} = (\mathbf{A} + [\mathbf{0}|\mathbf{TG}])^\top \cdot \mathbf{s}_0 + \mathbf{e}_0 \text{ and } \mathbf{c} = (\mathbf{A} + [\mathbf{0}|\mathbf{TG}])^\top \cdot \mathbf{s}_1 + \mathbf{e}_1 + [\mathbf{0}|\mathsf{ECC}(1)]$$

2. If $\|\mathbf{e}_0\| \leq s\sqrt{m}$ output $M' = 0$
3. If $\|\mathbf{e}_1\| \leq s\sqrt{m}$ output $M' = 1$
4. Output $\perp$

Correctness From Theorem 3.1, one easily verifies that if $\mathbf{c}$ is an encryption of $M \in \{0, 1\}$, then, $(\mathbf{s}_M, \mathbf{e}_M)$ as defined in the decryption algorithm above, verifies $(\mathbf{s}_M, \mathbf{e}_M) = (\mathbf{s}, \mathbf{e})$, where $(\mathbf{s}, \mathbf{e})$ are the random inputs of the encryption algorithm. In particular $\|\mathbf{e}_M\| = \|\mathbf{e}\| \leq s\sqrt{m}$ with overwhelming probability. It remains to check that we can't have both inequalities $\|\mathbf{e}_i\| \leq s\sqrt{m}$ holding at the same time. If it were the case, one would have

$$2 \cdot (\mathbf{A} + [\mathbf{0}|\mathbf{TG}])^\top (\mathbf{s}_0 - \mathbf{s}_1) + 2 \cdot (\mathbf{e}_0 - \mathbf{e}_1) = -2[\mathbf{0}|\mathsf{ECC}(1)] \bmod q$$

Since $2 \cdot \mathsf{ECC}(1) = 2\lfloor \frac{q}{2} \rfloor = \epsilon = 0$ or $-1 \bmod q$ this would imply the existence of two solutions to $(\mathbf{A}+[\mathbf{0}|\mathbf{TG}])^\top \mathbf{s}' + \mathbf{e}' = \mathbf{0}$, namely $(\mathbf{s}', \mathbf{e}') = (\mathbf{0}, \mathbf{0})$ and $(\mathbf{s}', \mathbf{e}') = (2(\mathbf{s}_0 - \mathbf{s}_1), 2(\mathbf{e}_0 - \mathbf{e}_1) + [\mathbf{0}|\epsilon])$. For both solutions $\|\mathbf{e}'\| \leq q \cdot \mathsf{negl}(\lambda)$, which contradicts Theorem 3.1. indeed this theorem implies that the function $(\mathbf{s}', \mathbf{e}') \in \mathbb{Z}_q^m \times D \mapsto (\mathbf{A}+[\mathbf{0}|\mathbf{TG}])^\top \mathbf{s}' + \mathbf{e}'$ is invertible, therefore injective for an error domain $D = \mathbb{Z}^m \cap r \cdot \mathfrak{B}$ up to an error radius $r = q/\operatorname{poly}(\lambda)$; while he have found a collision for a radius $r = q \cdot \mathsf{negl}(\lambda)$. In lattice terms, it would mean that the lattice spanned by the columns of $(\mathbf{A} + [\mathbf{0}|\mathbf{TG}])^\top$ contains a very short vector.

CCA-1/Tag-Based Security We recall that a CCA-1 scheme can easily be transformed to a CCA-2 scheme using a simple combination with a Strong OTS (see Section 2.3). In the following the CCA-2 scheme will be called MP and the CCA-1 scheme will be called MP*.

We now detail the security proof of the modified scheme.

Theorem 3.2 (Security of MP*) *MP* is selectively secure against chosen ciphertext attacks if* dLWE *holds.*

The proof is mostly similar to the one of the original scheme of [41]. It is given in appendix D.1.

3.2.2 Adding the SPHF

- HashKG: output a vector chosen from the Gaussian, $\mathsf{hk} = \mathbf{v} \leftarrow D^m_{\mathbb{Z},\omega(\sqrt{\log \lambda})}$, where $m = \bar{m} + nk$ as defined in Section 3.2.1.
- ProjKG($\mathsf{ek} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{G} \in \mathbb{Z}_q^{n \times nk}, \mathsf{hk} = \mathbf{v} \in \mathbb{Z}^m$): output $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$ where $\mathsf{hp}_1 = \mathbf{w}_1 = \mathbf{A} \cdot \mathbf{v} \in \mathbb{Z}_q^n, \mathsf{hp}_2 = \mathbf{w}_2 = [\mathbf{0}_{n \times \bar{m}}|\mathbf{G}] \cdot \mathbf{v} \in \mathbb{Z}_q^n$, where $\mathbf{0}_{n \times \bar{m}}$ is the zero matrix in $\mathbb{Z}_q^{n \times \bar{m}}$.
- ProjHash($\mathsf{hp} = \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{Z}_q^n, \mathsf{wit} = \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{T} \in \mathcal{T}$): output $H' = \mathsf{ECC}^{-1}(\langle \mathbf{w}_1, \mathbf{s} \rangle + \langle \mathbf{w}_2, \mathbf{T}^\top \mathbf{s} \rangle) \in \{0, 1\}$.
- Hash($\mathbf{c} \in \mathbb{Z}_q^m, \mathbf{T} \in \mathcal{T}, M', \mathsf{hk} = \mathbf{v}$): compute $\mathfrak{H} = \langle \mathbf{v}, \mathbf{c} - [\mathbf{0}|\mathsf{ECC}(M')] \rangle$, and output $H = \mathsf{ECC}^{-1}(\mathfrak{H}) \in \{0, 1\}$.

With this trick, a hk, hp can be computed before knowing the tag $\mathbf{T}$ and adapted on the fly once the tag is picked.

Correctness Once again, we only focus on valid encryption of $M' = M$, we have

$$\begin{aligned} H' &= \mathsf{ECC}^{-1}(\langle \mathbf{w}_1, \mathbf{s} \rangle + \langle \mathbf{w}_2, \mathbf{T}^\top \mathbf{s} \rangle) \\ &= \mathsf{ECC}^{-1}(\langle \mathbf{A} \cdot \mathbf{v}, \mathbf{s} \rangle + \langle [\mathbf{0}_{n \times \bar{m}}|\mathbf{G}]\mathbf{v}, \mathbf{T}^\top \mathbf{s} \rangle) \\ &= \mathsf{ECC}^{-1}(\langle \mathbf{v}, \mathbf{A}^\top \mathbf{s} \rangle + \langle \mathbf{v}, [\mathbf{0}_{n \times \bar{m}}|\mathbf{TG}]^\top \mathbf{s} \rangle) \\ &= \mathsf{ECC}^{-1}(\mathfrak{H} - \langle \mathbf{v}, \mathbf{e} \rangle) \end{aligned}$$

Notice that $\mathfrak{H}$ is uniformly random in $\mathbb{Z}_q$, while $\langle \mathbf{e}, \mathbf{v} \rangle \leq \|\mathbf{e}\| \|\mathbf{v}\| \leq s \cdot \omega(\sqrt{\log \lambda})m$, where the bound on $\|\mathbf{e}\|$ and $\|\mathbf{v}\|$ follows from Lemma 2.3. Therefore, by Lemma 2.5, $H' = \mathsf{ECC}^{-1}(\mathfrak{H} - \langle \mathbf{v}, \mathbf{e} \rangle) = H$ except with negligible probability.

Smoothness We will detail the case of a valid encryption of $M' \neq M$, the other one being relatively similar: If $M \neq M'$, then $\mathsf{ECC}(M) - \mathsf{ECC}(M') = \pm\lfloor \frac{q}{2} \rfloor$; therefore

$$H' - H = \mathsf{ECC}^{-1}\left(\langle(\mathbf{A} + [\mathbf{0}_{n\times\bar{m}}|\mathbf{TG}]) \cdot \mathbf{v}, \mathbf{s}\rangle\right) - \mathsf{ECC}^{-1}\left(\langle(\mathbf{A} + [\mathbf{0}_{n\times\bar{m}}|\mathbf{TG}]) \cdot \mathbf{v}, \mathbf{s}\rangle \pm \lfloor\frac{q}{2}\rfloor v_m + \langle \mathbf{v}, \mathbf{e}\rangle\right) = v_m \bmod 2$$

where the last equality holds except with negligible probability, following the same argumentation as above based on Lemmata 2.3 and 2.5. It remains to show that $v_m \bmod 2$ still has a uniform distribution over $\{0, 1\}$ despite the knowledge of $\mathsf{hp} = (\mathbf{w}_1, \mathbf{w}_2)$. First, one would rewrite

$$\mathbf{w}_1 = \bar{\mathbf{A}} \cdot (v_1 \dots v_{\bar{m}}) + (\mathbf{TG} - \mathbf{R}\bar{\mathbf{A}}) \cdot (v_{m-nk+1} \dots v_m)$$

and argue, using the leftover hash lemma that $\bar{\mathbf{A}} \cdot (v_1 \dots v_{\bar{m}})$ is uniformly random; therefore $\mathbf{w}_1$ does not reveal any information about the whole subvector $(v_{m-nk+1} \dots v_m)$. Next, we need to take a detailed look at $\mathbf{w}_2 = \mathbf{G} \cdot (v_{m-nk+1} \dots v_m)$ and especially at the gadget matrix as defined in [41]

$$\mathbf{G} = \begin{bmatrix} \cdots \mathbf{g}^\top \cdots & & & \\ & \cdots \mathbf{g}^\top \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g}^\top \cdots \end{bmatrix} \quad \text{where} \quad \mathbf{g}^\top = (1\ 2\ 4\ \dots 2^{k-1})$$

The only information about v_m is carried by the last coordinate w of $\mathbf{w}_2$

$$w = \underbrace{(1\ 2\ 4\ \dots 2^{k-2})}_{\bar{\mathbf{g}}^\top} \cdot \underbrace{(v_{m-k+1} \dots v_m - 1)^\top}_{\bar{\mathbf{v}}} + 2^{k-1} v_m \bmod q.$$

We are going to argue that $\bar{\mathbf{g}} \cdot \bar{\mathbf{v}}$ is almost uniform $\bmod\ q$; we recall that $q = 2^k$. Our argument is adapted from [41, Prop. 4.2]. Note that matrix $\mathbf{S}$ defined below is a basis of $\Lambda^\perp(\bar{\mathbf{g}}^\top)$:

$$\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 4 \end{bmatrix} \in \mathbb{Z}^{k-1\times k-1}.$$

Indeed, we have $\bar{\mathbf{g}}^\top \cdot \mathbf{S} = \mathbf{0} \bmod q$ and $\det(\mathbf{S}) = 2^k = q$; this implies that the smoothing parameter verifies

$$\eta_\epsilon(\Lambda^\perp(\bar{\mathbf{g}}^\top)) \leq \|\mathbf{S}\| \cdot \omega(\sqrt{\log \lambda}) = 4\omega(\sqrt{\log \lambda})$$

for some $\epsilon = \mathsf{negl}(\lambda)$ (see Definition A.9 and Lemma A.11). We conclude by Lemma A.10 that $\bar{\mathbf{g}}^\top \cdot \bar{\mathbf{v}}$ is negligibly close to uniform $\bmod\ q$. In particular $w = \bar{\mathbf{g}}^\top \cdot \bar{\mathbf{v}} + 2^{k-1} v_m$ does not reveal any information about v_m.

Last, by corollary A.12, $v_m \leftarrow D_{\mathbb{Z},\omega(\sqrt{\log \lambda})}$ taken modulo 2 is negligibly close to the uniform distribution $\mathcal{U}(\mathbb{Z}_2)$.

Remark Here we want to stress that it fulfils the classical Smoothness definition, however in the following we will need to take special care, because the Hash value is in $\{0, 1\}$, so even if an adversary has no advantage in finding the Hash Value (the best attack is picking at random), he can still manage to do it with probability 0.5. However, as we are going to do a linear number $\mathfrak{n}$ of SPHF and do a concatenation of them the probability that all of them are fulfilled goes down to $0.5^{\mathfrak{n}}$ if the adversary tries to guess at random.

Pseudo-Randomness Once again, under the indistinguishability of the encryption, we can transform a commitment to a word in the language, to a commitment to a word outside of the language, hence an adversary against the Pseudo-Randomness either break the Smoothness (which, information theoretically he can not) or the indistinguishability of the encryption.

4 Password Authenticated Key-Exchange

The only existing lattice-based construction of PAKE is given in [36, 23]. It is a 3-round protocol, proven in a variant of the BPR model [7]. We give two improvements of this protocol in this section. First, we give an equivalent 3-round protocol, but this time proven in the UC framework [15, 18]. To do so, we are going to use the homomorphic properties on the randomness in the CCA-1 encryption from [41], upgraded to CCA-2 with a strong one-time signature and the previous home-made smooth projective hash functions. Secondly, we give a one-round protocol in the BPR model [7] by taking advantage of the strong properties of the SPHF we present in Section 3.2.2. The projection key can indeed be computed before having seen the corresponding commitment, which enables us to get rid of the other rounds.

$\mathsf{Commit}(\mathfrak{L}, \mathsf{ek}, M, N; \mathbf{r}, \mathbf{s}) :$ for $(\mathbf{r}, \mathbf{s}) \xleftarrow{\$} (\mathbb{Z}_p^n)^2$

$(\mathcal{C}, \mathcal{C}') \leftarrow \mathsf{DMPCom}(\mathfrak{L}, \mathsf{ek}, M, 0; \mathbf{r}, \mathbf{s})$

$\chi = \mathfrak{H}_K{}'(\mathcal{C}'), \mathcal{C}'' = \mathsf{Trap.Commit}(\chi; \mathbf{t})$ $\xrightarrow{\mathcal{C}, \mathcal{C}''}$

$\zeta \stackrel{?}{\neq} \mathbf{0}$ $\xleftarrow{\zeta}$ $\zeta \leftarrow D^{\mathfrak{n}}_{\mathbb{Z},\omega(\sqrt{\log \lambda})}$

$\mathbf{z} = \mathbf{r} + \zeta \mathbf{s}$

$\mathsf{Decommit}(\mathfrak{L}, \mathcal{C}, \mathcal{C}', \zeta) :$ $\xrightarrow{\mathcal{C}', \mathbf{t}}$ $\mathsf{Trap.Open}(\mathsf{ck}, \mathcal{C}'', \mathfrak{H}_K{}'(\mathcal{C}'), \mathbf{t})?$

Implicit check with a SPHF that $\mathsf{MP}^*(\mathfrak{L}, \mathsf{ek}, M, \mathbf{T}; \mathbf{z}) \stackrel{?}{\approx} \mathcal{C} + \epsilon \mathcal{C}'$

Figure 4: DMPCom' Commitment Scheme for SPHF

4.1 Building Blocks for our UC PAKE

In a nutshell, when dealing with PAKE schemes constructed from the KOY-GL approach (with each player sending a commitment and a projection key for an SPHF), the UC framework requires that the first commitment sent is simultaneously extractable (in case it is sent by the adversary, the simulator needs to extract the value committed to) and equivocable (in case it is sent by an honest player, the simulator needs to be able to change its mind later on). The first solution was given in [18] but requires another commitment sent in a pre-flow and the use of a simulation-sound non-interactive zero-knowledge proof to ensure that the values committed to are the same in both flows. More recently, it was shown that it is indeed possible to construct a commitment both extractable and equivocable [2, 39, 8]. We follow their approach here.

Multi-Bits Micciancio Peikert Encryption. One can extend the encryption scheme MP to vectors of k bits. We will denote it k-MP. It consists in encrypting each bit with independent random coins in $\mathcal{C}_b$ but a shared Tag $\mathbf{T} = \mathfrak{H}_K(\mathfrak{L}, (\mathsf{vk}_b)_{b \in [\![1,k]\!]})$. For readability, we will omit vk in the following as we don't mention the one-time signature but it is crucial to remember it is part of the Tag to provide CCA-2.

Double Micciancio Peikert Commitment. To fulfill the equivocability on the first commitment, we need to be able to give a "doubled" variant of our revision of the Micciancio Peikert (Bit-)Encryption.[4] See the Appendix B for more details, but we briefly present here the commitment scheme that we will use in the rest of this paper in conjunction with SPHF.

To make it equivocable, we double the commitment process, in two steps. The CRS additionally contains an encryption key for a trapdoor commitment ck (the trapdoor for equivocability is then called $\aleph$, see Section A.2.1 for more details).

The Double Micciancio Peikert Encryption encryption scheme, denoted DMP and detailed in the Appendix B is

$$\mathsf{DMP}(\mathfrak{L}, \mathsf{ek}, M, N; \mathbf{r}, \mathbf{s}) \stackrel{\text{def}}{=} (\mathcal{C} \leftarrow \mathsf{MP}(\mathfrak{L}, \mathsf{ek}, M; \mathbf{r}), \mathcal{C}' \leftarrow \mathsf{MP}^*(\mathfrak{L}, \mathsf{ek}, N, \mathbf{T}; \mathbf{s}))$$

where $\mathbf{T} = \mathfrak{H}_K(\mathfrak{L}, \mathsf{vk})$ is computed during the generation of $\mathcal{C}$ and transferred for the generation of $\mathcal{C}'$. The difference between MP and MP^* resides in the fact we don't need to include the Strong One-Time Signature to the MP^* version.

We will use DMPCom to denote the use of DMP with the encryption key ek. The usual commit/decommit processes are described on Figure 8 in the Appendix B. We clearly cannot do the decommit phase (otherwise we reveal the password used), so that we need a variant of this commitment, where one can implicitly check the opening with an SPHF. This DMPCom' scheme can be found on Figure 4. We stress that at this stage, we use $\chi = \mathfrak{H}_K{}'(\mathcal{C}')$ (for the SPHF implicit check), later in this paper (see Section 5), when we present our UC Commitment, we use instead $\chi = \mathfrak{H}_K{}'(M, \mathcal{C}')$ (for the explicit check).

The DMPCom' scheme in this way is not formally extractable/binding: in DMPCom', the sender can indeed encrypt M in $\mathcal{C}$ and $N \neq 0$ in $\mathcal{C}'$, and then, the global ciphertext $\mathcal{C} + \zeta \mathcal{C}'$ contains $M' = M + \zeta N \neq M$, whereas one would have extracted M from $\mathcal{C}$. But M' is unknown before ζ is sent, and thus, if one checks the membership of M' to a sparse language, it will unlikely be true. However, contrary to [39, 8], we do not consider a sparse language here (they consider the language reduced to one element of a group of big order q). Rather, we here consider a language of bits, making the adversary manage to find the correct value of M' with probability $1/2$. A simple solution is to make the verifier send a number $\mathfrak{n}$ of challenges ζ linear in the security parameter. This way, the probability becomes $1/2^{\mathfrak{n}}$, which is negligible. The implicit verification can still easily be done using a conjunction of SPHF, constructed by standard techniques [2, 8] (this verification fails as soon as one verification fails). For the sake of simplicity of the notations, even if one has to keep in mind the use of these multiple challenges, we continue to denote as ζ, $\mathcal{C}'$ and $\mathbf{z}$ the list of the corresponding values.

Multi-Bits Double Micciancio Peikert Commitment. One can extend these encryption and commitment schemes to vectors of k bits (see the Appendix B). We will denote them k-DMPCom' or k-DMPCom for the commitment schemes. They consist in encrypting each bit with independent random coins in $\mathcal{C}_b$ but a shared Tag $\mathbf{T} = \mathfrak{H}_K(\mathfrak{L}, (\mathsf{vk}_b)_{b \in [\![1,k]\!]})$, together with independent companion ciphertexts $\mathcal{C}'_{b,j}$ of 0, still with the same tag for the doubled version. In the latter case, $\mathfrak{n}$ independent challenges $\zeta_{b,j}$ are then sent for each bit of the message, to lead to the full commitments $\{\mathcal{C}_b + \zeta_{b_j} \mathcal{C}'_{b,j}\}_{b,j}$. For the sake of

[4] This is an artificial tool to allow both extraction and equivocation. Indeed, any IND-CCA labeled encryption scheme can be used as a non-malleable and extractable labeled commitment scheme. In order to add the equivocability, one can use a technique inspired from [39, 8].

P_i uses a password W_i and P_j uses a password W_j. We denote $\mathfrak{L} = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.

- First Round: P_i (with random tape ω_i) first generates a pair of signing/verification keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ and a k-DMPCom' commitment on W_i in $(\mathcal{C}_i, \mathcal{C}'_i)$, with randomness $(\mathbf{r}_i, \mathbf{r}'_i)$, under $\mathfrak{L}_i = (\mathfrak{L}, \mathsf{VK}_i)$: $(\mathcal{C}_i, \mathcal{C}'_i) = k\text{-}\mathsf{DMPCom}'(\mathfrak{L}_i, \mathsf{ek}, W_i, 0; \mathbf{r}_i, \mathbf{r}'_i)$. It also computes a Trapdoor commitment on $\mathcal{C}'_i$ in $\mathcal{C}''_i$ (with randomness $\mathbf{t}$): $\chi = \mathfrak{H}_K{}'(\mathcal{C}'_i)$ and $\mathcal{C}''_i = \mathsf{Trap.Commit}(\chi; \mathbf{t})$. It then sends $(\mathsf{VK}_i, \mathcal{C}_i, \mathcal{C}''_i)$ to P_j;
- Second Round: P_j (with random tape ω_j) computes a k-MP encryption on W_j in $\mathsf{Com}_j = \mathcal{C}_j$, with witness $\mathbf{r}_j$, under the label $\mathfrak{L}_j = \mathfrak{L}$: $\mathcal{C}_j = k\text{-}\mathsf{MP}(\mathfrak{L}_j, \mathsf{ek}, W_j; \mathbf{r}_j)$. It then generates a challenge ζ on $\mathcal{C}_i$ and hashing/projection keys hk_i and the corresponding hp_i for the equality test on Com_i: "Com_i is a valid commitment of W_j". It then sends $(\mathcal{C}_j, \zeta, \mathsf{hp}_i)$ to P_i;
- Third Round: user P_i can compute $\mathsf{Com}_i = \mathcal{C}_i + \zeta\mathcal{C}'_i$ and witness $\mathbf{z} = \mathbf{r}_i + \zeta\mathbf{r}'_i$, it generates hashing/projection keys hk_j and hp_j for the equality test on Com_j. It finally signs all the flows using SK_i in σ_i and sends $(\mathcal{C}'_i, \mathbf{t}, \mathsf{hp}_j, \sigma_i)$ to P_j;
- Hashing: P_j first checks the signature and the validity of the Trapdoor commitment (thanks to $\mathbf{t}$), it computes $\mathsf{Com}_i = \mathcal{C}_i + \zeta\mathcal{C}'_i$. P_i computes K_i and P_j computes K_j as follows:

$$\begin{aligned} K_i &= \mathsf{Hash}(\mathsf{hk}_j, \mathrm{L}'_j, \mathfrak{L}, \mathsf{Com}_j) \oplus \mathsf{ProjHash}(\mathsf{hp}_i, \mathrm{L}_i, \mathfrak{L}_i, \mathsf{Com}_i; \mathbf{z}) \\ K_j &= \mathsf{ProjHash}(\mathsf{hp}_j, \mathrm{L}_j, \mathfrak{L}, \mathsf{Com}_j; \mathbf{r}_j) \oplus \mathsf{Hash}(\mathsf{hk}_i, \mathrm{L}'_i, \mathfrak{L}_i, \mathsf{Com}_i) \end{aligned}$$

Figure 5: Password-based Authenticated Key Exchange

simplicity of the notations, we continue to denote by ζ and $\mathcal{C}'$ the list of those values $\zeta_{b,j}$ and $\mathcal{C}'_{b,j}$ in the description of our PAKE protocol. Again, if one of the companion ciphertext $\mathcal{C}'_{b,j}$ does not encrypt 0, the full commitment encrypts a vector with at least one unpredictable component M'_b (with the trick explained in the former paragraph, since we use a linear number of challenges). Several non-unity components in the companion ciphertexts would lead to independent components in the full commitment. For languages sparse enough, this definitely turns out not to be in the language. The implicit verification is done with the conjunction of the SPHF for every bit, see details in Section 3.2.2. Again, for the sake of simplicity of the notations of our PAKE protocol, we only mention the global SPHF used, omitting the details of its construction.

4.2 A Three-Round PAKE with low communication proven in the UC framework

4.2.1 Functionality of Password-Authenticated Key Exchange

We use the functionality of PAKE presented in [18]. We recall it in Appendix C for lack of space.

4.2.2 Construction of our Password-Authenticated Key Exchange Protocol

We use the generic construction given in [8], as described on Figure 5. This protocol consists in three rounds, the first and the third one being a double Micciancio Peikert commitment sent by P_i. We thus need this player to generate a pair of one-time signature keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ in order to sign his second flow and avoid man-in-the-middle attacks. Since P_j is the second player to send his flow, a simple Micciancio Peikert commitment, as presented in Section 3.2.1 is enough. Both players also send a projection key to each other, in order to check the validity of these commitments, and thus that they share the same password. This will lead to a session key constructed from the two (projected) hash values. Note that the projection key needs to be sent by P_j before having sent the entire Com_i sent by P_i, but the SPHF indeed only depends on $\mathcal{C}_i$, already known by P_j. Furthermore, the SPHF constructed in Section 3.2.2 has the nice property that the projection key does not depend on the commitment. This is a stronger property than the one presented in [36]. Speaking in terms of languages to be consistent with the presentation of our SPHF in Section 3.2.2, P_i uses a password W_i and expects P_j to own the same password, i.e. a word in the language $\mathrm{L}'_j = \mathrm{L}_i = \{W_i\}$. Similarly, P_j uses a password W_j and expects P_i to own the same password, i.e. a word in the language $\mathrm{L}'_i = \mathrm{L}_j = \{W_j\}$.

Theorem 4.1 *Our PAKE scheme from Figure 5 realizes the $\mathcal{F}_{\mathrm{pwKE}}$ functionality in the $\mathcal{F}_{\mathrm{crs}}$-hybrid model, in the presence of static adversaries, under the* LWE *and* SIS *assumptions and the security of the One-Time Signature.*

Technically, one can use a CPA encryption for $\mathcal{C}_j$, so for optimization purpose it would be better to use the Dual Regev Encryption and its associated SPHF in this case, however for a symmetrical approach in design, we use MPon both sides.

4.2.3 Extension to LAKE

Recent works [8, 9] also used SPHF to build protocols more modular than classical PAKE, where they do not restrain the languages to only words. This primitive, called LAKE for Language Authenticated Key Exchange, allows two users, Alice

and Bob, each owning a word in a specific language, to agree on a shared high entropy secret if each user knows a word in the language the other thinks about. This notion supersedes PAKE, verifier-based PAKE, Secret Handshakes, CAKEs, ...

In our case, due to the bit per bit nature of the commitment we can consider only languages of ciphertexts on bitstrings. Due to the algebraic nature of SPHF, one can use classical techniques to extend this to any conjunction or disjunction of languages, allowing this way to have advanced-PAKE protocols that handle any polynomial number of conjunctions and disjunctions of languages of the form $\mathcal{L}_{\mathcal{I},\mathcal{J}} = \{w \in \{0,1\}^n | \forall i \in \mathcal{I}, w_i = 0, \forall j \in \mathcal{J}, w_j = 1\}$.

4.3 A One-Round PAKE

As explained earlier, Katz and Vaikuntanathan [36] recently proposed the first PAKE protocol based on lattices, using a general framework following KOY-GL's approach [35, 28]. As in their article, we consider here the standard notions of security [7, 35, 28] but we improve the efficiency of their construction by giving a one-round protocol.

The high-level idea follows that of [37, 9]: In the KOY-GL framework, each player sends an encryption of the password, and then uses an SPHF on the partner's ciphertext to check whether it actually contains the same password. The two hash values are multiplied to produce the session key. If the encrypted passwords are the same, the different ways to compute the hash values (Hash and ProjHash) give the same results. If the passwords differ, the smoothness makes the values computed by each player independent. To this aim, the authors need an SPHF on a labeled IND-CCA encryption scheme.

Moreover, to allow a one-round PAKE, the ciphertext and the projection key on the partner's ciphertext should be sent together, before having seen the partner's ciphertext: the projection key should thus be independent of the ciphertext. In addition, the adversary can wait to have received the partner's projection key before generating the ciphertext, and thus a stronger smoothness is required. This is exactly why we need a strong type of SPHF, as constructed in Section 3.2.2, in the one-round PAKE framework. The protocol is presented on Figure 6.

P_i		P_j
$\mathcal{C}_i : k\text{-}\mathsf{MP}(\mathsf{ek}, W_i; r_i)$		$\mathcal{C}_j : \mathsf{MP}(\mathsf{ek}, W_j; r_j)$
$\mathsf{hk}_i \leftarrow_\$, \mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathcal{L}_i, \mathsf{hk}_i)$		$\mathsf{hk}_j \leftarrow_\$, \mathsf{hp}_j \leftarrow \mathsf{ProjKG}(\mathcal{L}_j, \mathsf{hk}_j)$
	$\xleftarrow[\mathsf{hp}_i, \mathcal{C}_i, \sigma_i]{\mathsf{hp}_j, \mathcal{C}_j, \sigma_j}$	
$\mathsf{Verify}(\sigma_j)$		$\mathsf{Verify}(\sigma_i)$
$H'_j : \mathsf{ProjHash}(\mathsf{hp}_j, \mathcal{C}_i, w_i)$		$H'_i : \mathsf{ProjHash}(\mathsf{hp}_i, \mathcal{C}_j, w_j)$
$H_i : \mathsf{Hash}(\mathsf{hk}_i, \mathcal{C}_j)$		$H_j : \mathsf{Hash}(\mathsf{hk}_j, \mathcal{C}_i)$
$s_i : H_i \oplus H'_j$		$s_j : H'_i \oplus H_j$

Figure 6: One-Round Pake

Theorem 4.2 *Our one-round PAKE scheme from Figure 6 is secure in the BPR model, under the* LWE *and* SIS *assumptions.*

Proof: Since the scheme exactly follows the scheme given in [9], we only have to check that our primitives (MP encryption and SPHF) fulfill the same properties in order to be able to apply modularly the proof given in [9, Theorem 4]. This was done in Sections 3.2.1 and 3.2.2, which concludes the proof. ∎

5 UC Commitment

On Figure 7 below, we will provide the first UC commitment based on Lattices. This scheme is inspired by the ideas from [39, 10]. We will consider a bit commitment, and using techniques explained before, this can easily be generalized to any bitstring.

Functionality of Commitment. We use the functionality of Commitment presented in [16, 39]. We recall it in Appendix C for lack of space.

Description of the Protocol. The commit phase requires only 3 rounds, and the decommit is straightforward.

As explained when we introduced the DMP scheme in Section 4, we are now going to include the value x in the second Trapdoor Commitment to prevent the adversary from trying to open his commitment to another value. We stress again that the verification of the strong one time signature is implied every time someone receives a MP encryption. This is required to be able to rely on the IND-CCA-2 property in the simulation. We only describe the adaptive version of this protocol, one can easily switch commitment rounds to the decommitment to obtain a fair comparison with Lindell's protocol with static corruption. We describe the version on one bit, once again we could just use multiple commitment to extend to any message.

We have a CRS, consisting of $(\mathsf{ck}, \mathsf{ek}, \mathfrak{H}_K, \mathfrak{H}_K{}')$, respectively the commitment key for a trapdoor commitment, the encryption key for a Micciancio Peikert like scheme, and randomly drawn from a collision-resistant hash function family $\mathcal{H}$, one arriving in the tagspace, the other in the bitstring space.

The commit phase. Upon receiving a message $(\mathsf{Commit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$, party P_i works as follows, where $x \in \{0,1\}$ and $\mathsf{sid}, \mathsf{ssid} \in \{0,1\}^{\log^2(n)/4}$.

1. P_i picks $\mathbf{r}, \mathbf{s} \leftarrow \mathbb{Z}_p^n$ and computes $(C_1, C_2) = \mathsf{DMP}(x, 0; \mathbf{r}, \mathbf{s}, \mathbf{T} = \mathfrak{H}_K(\mathsf{sid}, \mathsf{ssid}, \mathsf{vk}))$ with noise $\mathbf{e}_r, \mathbf{e}_s$.
   P_i picks $\mathbf{t}_1, \mathbf{t}_2 \leftarrow D_{\mathbb{Z}^m, s}$.
   He computes $c_t^1 = \mathsf{Trap.Commit}(\xi, (C_1); \mathbf{t}_1)$, $c_t^2 = \mathsf{Trap.Commit}(\xi, (x, C_2, \mathsf{sid}, \mathsf{ssid}, P_i, P_j); \mathbf{t}_2)$.
   He sends (c_t^1, c_t^2) to P_j.
2. P_j picks $\zeta \leftarrow D^*_{\mathbb{Z}, \omega(\sqrt{\log \lambda})}$ and sends it to P_i.
3. P_i now computes $\mathbf{z} = \mathbf{r} + \zeta \mathbf{s}, \mathbf{e}_z = \mathbf{e}_r + \zeta \mathbf{e}_s$, and erases $\mathbf{r}, \mathbf{s}, \mathbf{e}_r, \mathbf{e}_s$.
   He also opens c_t^1 by sending $(C_1, \mathbf{t}_1)$ to P_j.
4. P_j verifies the consistency of c_t^1 using $\mathsf{Trap.Open}(\mathsf{ck}, C_1, c_t^1, \mathbf{t}_1)$.
   If yes, he stores $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, C_1, \zeta, \mathbf{e}_z, c_p^2)$ and outputs $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.
   He ignores any later commitment messages with the same $(\mathsf{sid}, \mathsf{ssid})$ from P_i.

The decommit phase. Upon receiving a message $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$, P_i works as follows:

1. P_i sends $(x, C_2, \mathbf{t}_2, \mathbf{z}, \mathbf{e}_z)$ to P_j.
2. P_j outputs $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$ if and only if c_t^2 is consistent and $\mathbf{e}_z$ is of appropriate size and:
$$\mathsf{MP}(\mathsf{ek}, x; \mathbf{T}, \mathbf{z}, \mathbf{e}_z) = C_1 + \zeta C_2$$

Figure 7: Our New Commitment Protocol UC-Secure against Adaptive Adversaries

Theorem 5.1 *Our Commitment scheme from Figure 7 realizes the $\mathcal{F}_{\mathsf{mcom}}$ functionality in the $\mathcal{F}_{\mathrm{crs}}$-hybrid model, in the presence of adaptive adversaries, under the* LWE *and* SIS *assumptions.*

The complete proof, and the associated simulator can be found in the Appendix D.4.

References

[1] M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, and D. Pointcheval. SPHF-Friendly Non-Interactive Commitment Schemes. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - Proceedings of ASIACRYPT '13*, Lecture Notes in Computer Science, Bangalore, India, Dec. 2013. Springer. To appear. 2

[2] M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, Aug. 2009. 1, 2, 10, 18, 27

[3] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996. 3

[4] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *CRYPTO (1)*, pages 57–74, 2013. 4

[5] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993. 3

[6] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, Apr. 2012. 4, 5

[7] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000. 1, 2, 9, 12

[8] F. Ben Hamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Efficient uc-secure authenticated key-exchange for algebraic languages. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 272–291. Springer, 2013. 2, 5, 10, 11, 22, 27

[9] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New Techniques for SPHFs and Efficient One-Round PAKE Protocols. In R. Canetti and J. Garay, editors, *Advances in Cryptology - Proceedings of CRYPTO '13*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475, Santa Barbara, California, Aug. 2013. Springer. `http://eprint.iacr.org/2013/034`. 2, 11, 12

[10] O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Analysis and Improvement of Lindell's UC-Secure Commitment Schemes. In R. Safavi-Naini and M. E. Locasto, editors, *Conference on Applied Cryptography and Network Security (ACNS '13)*, volume 7954 of *Lecture Notes in Computer Science*, pages 534–551, Banff, Alberta, Canadan, June 2013. Springer. 2, 12

[11] O. Blazy, D. Pointcheval, and D. Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In R. Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111. Springer, Mar. 2012. 1

[12] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society Press, Oct. 2011. 4

[13] Z. Brakerski and V. Vaikuntanathan. Lattice-based fhe as secure as pke. Cryptology ePrint Archive, Report 2013/541, 2013. 4

[14] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, Aug. 2003. 2

[15] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, Oct. 2001. 2, 9, 18, 23, 24

[16] R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, Aug. 2001. 2, 12

[17] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, May 2004. 5, 17

[18] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, May 2005. 2, 9, 10, 11, 18, 24, 27

[19] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503. ACM Press, May 2002. 2

[20] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, May 2010. 19

[21] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, Apr. / May 2002. 1, 5, 18, 33

[22] I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer, Aug. 2002. 2

[23] Y. Ding and L. Fan. Efficient password-based authenticated key exchange from lattices. *2012 Eighth International Conference on Computational Intelligence and Security*, 0:934–938, 2011. 2, 9

[24] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, Aug. 1985. 33

[25] M. Fischlin, B. Libert, and M. Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 468–485. Springer, Dec. 2011. 2

[26] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013. 1

[27] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO (2)*, pages 479–499, 2013. 1

[28] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`. 1, 12, 18

[29] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009. 1

[30] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008. 3, 20, 33

[31] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 545–554, New York, NY, USA, 2013. ACM. 1

[32] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, Apr. 2008. 2

[33] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 20

[34] Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95. Springer, May 2005. 18

[35] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, May 2001. 1, 12

[36] J. Katz and V. Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 636–652. Springer, Dec. 2009. 2, 6, 9, 11, 12

[37] J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Y. Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer, Mar. 2011. 2, 5, 12

[38] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, Mar. 2006. 5, 16, 17

[39] Y. Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 446–466. Springer, May 2011. 2, 10, 12, 18, 21, 24

[40] V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In R. Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, Mar. 2008. 19

[41] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, Apr. 2012. 7, 8, 9, 19, 21

[42] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381. IEEE Computer Society Press, Oct. 2004. 3, 20

[43] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, Aug. 2008. 2

[44] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005. 3, 4

A Security Notions and Hardness Assumptions

A.1 Security Definition

Public Key Encryption Scheme. A public key encryption scheme $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ is defined as follows:

- $(\mathsf{ek}, \mathsf{dk}) \leftarrow_{\$} \mathsf{KG}(1^n)$: the randomized key generation algorithm with security parameter $n \in \mathbb{N}$;
- $c \leftarrow_{\$} \mathsf{Enc}(\mathsf{ek}, m)$: the randomized encryption algorithm encrypts the message $m \in \mathcal{M}$ to get a ciphertext C;
- $m \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$: the (deterministic) decryption algorithm decrypts the ciphertext c to get back the plaintext or a rejection symbol.

PKE is perfectly correct if for any $n \in \mathbb{N}$, all $(\mathsf{ek}, \mathsf{dk}) \leftarrow_{\$} \mathsf{KG}(1^n)$ and all messages $m \in \mathcal{M}$ we have $\Pr[\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m)) = m] = 1$.

Definition A.1 (Security of Public Key Encryption) *Public key encryption scheme* PKE *is* (τ, ε, Q)*-secure against adaptive chosen ciphertext attacks (CCA-2) if and only if*

$$|\Pr[\mathbf{Exp}^{\mathsf{CCA\text{-}2}}_{\mathsf{TBE},\mathcal{A},Q}(n) = 1] - \frac{1}{2}| \leq \varepsilon$$

holds for any PPT adversary $\mathcal{A}$ *with running time* τ*, where* $\mathbf{Exp}^{\mathsf{CCA\text{-}2}}_{\mathsf{PKE},\mathcal{A},Q}(n)$ *is defined in Table 1.* $\mathtt{ODec}(c)$ *is an oracle returns* $m \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$ *and* $\mathcal{A}$ *can only query* $\mathtt{ODec}(\cdot)$ *at most* $Q = \mathrm{poly}(n)$ *times.*

Specially, there are two weaker notions on the public key encryption: if $Q = 0$*, then we say the scheme is secure against chosen plaintext attacks (CPA); if the adversary* $\mathcal{A}$ *is not allowed to query* $\mathtt{ODec}(\cdot)$ *after seeing the challenge ciphertext* c^**, then we say the scheme is secure against non-adaptive chosen ciphertext attacks (CCA-1).*

Experiment $\mathbf{Exp}^{\mathsf{CCA\text{-}2}}_{\mathsf{PKE},\mathcal{A},Q}(n)$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow_{\$} \mathsf{KG}(1^n)$;
$(m_0, m_1, St) \leftarrow_{\$} \mathcal{A}^{\mathtt{ODec}(\cdot)}(\mathsf{ek})$;
$b \leftarrow_{\$} \{0, 1\}$; $c^* \leftarrow_{\$} \mathsf{Enc}(\mathsf{ek}, m_b)$
$b' \leftarrow_{\$} \mathcal{A}^{\mathtt{ODec}(\cdot)}(c^*, St)$;
If $b' = b$ then return 1 else return 0.

Table 1: Security Experiment for Public Key Encryption

Tag-based Encryption Scheme. A tag-based encryption [38] is similar to the public key encryption except that the encryption and decryption operations take an additional tag, which is a binary string of appropriate length and need not have any particular internal structure. Formally, a tag-based encryption scheme $\mathsf{TBE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ with tag space $\mathcal{T}$ and message space $\mathcal{M}$ is defined as follows:

- $(\mathsf{ek}, \mathsf{dk}) \leftarrow_{\$} \mathsf{KG}(1^n)$: the randomized key generation algorithm with security parameter $n \in \mathbb{N}$;
- $c \leftarrow_{\$} \mathsf{Enc}(\mathsf{ek}, t, m)$: the randomized encryption algorithm encrypts the message $m \in \mathcal{M}$ with tag $t \in \mathcal{T}$ to get a ciphertext C;
- $m \leftarrow \mathsf{Dec}(\mathsf{dk}, t, c)$: the (deterministic) decryption algorithm decrypts the ciphertext c with tag t to get back the plaintext or a rejection symbol.

TBE is perfectly correct if for any $n \in \mathbb{N}$, all $(\mathsf{ek}, \mathsf{dk}) \leftarrow_{\$} \mathsf{KG}(1^n)$, all tags t and messages $m \in \mathcal{M}$ we have $\Pr[\mathsf{Dec}(\mathsf{dk}, t, \mathsf{Enc}(\mathsf{ek}, t, m)) = m] = 1$.

Definition A.2 (Security of Tag-based Encryption) *Tag-based encryption scheme* TBE *is* (τ, ε, Q)*-selective-tag weak security against chosen ciphertext attacks (*$\mathsf{tbe\text{-}stag\text{-}cca}$*) if and only if*

$$|\Pr[\mathbf{Exp}^{\mathsf{tbe\text{-}stag\text{-}cca}}_{\mathsf{TBE},\mathcal{A},Q}(n) = 1] - \frac{1}{2}| \leq \varepsilon$$

holds for any PPT adversary $\mathcal{A}$ *with running time* τ*, where* $\mathbf{Exp}^{\mathsf{tbe\text{-}stag\text{-}cca}}_{\mathsf{TBE},\mathcal{A},Q}(n)$ *is defined in Table 2.* $\mathtt{ODec}(t, c)$ *is an oracle returns* $m \leftarrow \mathsf{Dec}(\mathsf{dk}, t, c)$ *with restriction that* $\mathcal{A}$ *is not allowed to query with target tag* t^**.* $\mathcal{A}$ *can only query* $\mathtt{ODec}(\cdot, \cdot)$ *at most* $Q = \mathrm{poly}(n)$ *times.*

Experiment $\mathbf{Exp}^{\mathsf{tbe\text{-}stag\text{-}cca}}_{\mathsf{TBE},\mathcal{A},Q}(n)$
$t^* \leftarrow_{\$} \mathcal{A}(1^n)$;
$(\mathsf{ek}, \mathsf{dk}) \leftarrow_{\$} \mathsf{KG}(1^n)$;
$(m_0, m_1, St) \leftarrow_{\$} \mathcal{A}^{\mathtt{ODec}(\cdot,\cdot)}(\mathsf{ek})$;
$b \leftarrow_{\$} \{0,1\}$; $c^* \leftarrow_{\$} \mathsf{Enc}(\mathsf{ek}, t^*, m_b)$
$b' \leftarrow_{\$} \mathcal{A}^{\mathtt{ODec}(\cdot,\cdot)}(c^*, St)$;
If $b' = b$ then return 1 else return 0.

Table 2: Security Experiment for Tag-based Encryption

As shown in [38, 17], tbe-stag-cca encryption scheme can be transferred into a CCA-2 public key encryption by using a Strong One-time Signature.

Trapdoor Commitment. A commitment is defined as $\mathsf{COM} = (\mathsf{CKG}, \mathsf{Commit}, \mathsf{Open})$:

- $(\mathsf{ck}, \mathsf{td}) \leftarrow_{\$} \mathsf{CKG}(1^n)$: the randomized key generation algorithm outputs the commitment key ck and the trapdoor td;
- $(c, r) \leftarrow_{\$} \mathsf{Commit}(\mathsf{ck}, m)$: the randomized commitment algorithm inputs the commitment key ck and the message m and then it outputs the commitment value c and the opening information w;
- $1/0 \leftarrow \mathsf{Open}(\mathsf{ck}, m, c, r)$: the deterministic opening algorithm decommits the commitment value c with the opening information r. If c is indeed the commitment of m then it outputs 1; otherwise, it outputs 0.

Definition A.3 (Trapdoor Commitment) *A commitment scheme* COM *with message space $\mathcal{M}$ is called trapdoor commitment if it satisfies the followings:*

- **Perfect hiding.** *This property states the following two distributions are identical:*

$$\{c_0 : (\mathsf{ck}, \mathsf{td}) \leftarrow_{\$} \mathsf{CKG}(1^n), (c_0, r_0) \leftarrow_{\$} \mathsf{Commit}(\mathsf{ck}, m_0)\}$$

$$\{c_1 : (\mathsf{ck}, \mathsf{td}) \leftarrow_{\$} \mathsf{CKG}(1^n), (c_1, r_1) \leftarrow_{\$} \mathsf{Commit}(\mathsf{ck}, m_1)\}$$

  where $m_0 \neq m_1$.

- **Computational binding.** COM *is (τ, ε)-binding if the following holds for any PPT adversary $\mathcal{A}$ with running time τ:*

$$\Pr\begin{bmatrix} (\mathsf{ck}, \mathsf{td}) \leftarrow_{\$} \mathsf{CKG}(1^n); ((m_0, c, r_0), (m_1, c, r_1)) \leftarrow_{\$} \mathcal{A}(\mathsf{ck}) : \\ \mathsf{Open}(\mathsf{ck}, m_0, c, r_0) = \mathsf{Open}(\mathsf{ck}, m_1, c, r_1) = 1 \wedge (m_0, r_0) \neq (m_1, r_1) \end{bmatrix} \leq \varepsilon.$$

- **Perfect trapdoor opening.** *There exist an efficient algorithm* Topen *given the trapdoor* td *can open the commitment to any message. Formally, the following holds:*

$$\Pr\begin{bmatrix} (\mathsf{ck}, \mathsf{td}) \leftarrow_{\$} \mathsf{CKG}(1^n); m_0, m_1 \leftarrow_{\$} \mathcal{M}; (c_0, r_0) \leftarrow_{\$} \mathsf{Commit}(\mathsf{ck}, m_0); \\ r_1 \leftarrow_{\$} \mathsf{Topen}(\mathsf{td}, (m_0, r_0), m_1) : \mathsf{Open}(\mathsf{ck}, m_1, c_0, r_1) = 1 \wedge m_0 \neq m_1 \end{bmatrix} = 1.$$

One time Signature A signature scheme SIG with message space $\mathcal{M}$ is defined as a triple of probabilistic polynomial time (PPT) algorithms $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$:

- $(\mathsf{vk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(1^n)$: the randomized key generation algorithm takes as an input the unary representation of the security parameter 1^n and outputs a verification key vk and signing key sk.
- $s \leftarrow_{\$} \mathsf{Sign}(\mathsf{sk}, m)$: the randomized signing algorithm takes as input a signing key sk and message m and outputs a signature s.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{vk}, m, s)$: the deterministic verification algorithm takes as input a verification key vk and a message-signature pair (m, s) outputs 1 (accept) or 0 (reject).

SIG is perfectly correct if for any $n \in \mathbb{N}$, all $(\mathsf{vk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(1^n)$, all $m \in \mathcal{M}$, and all $s \leftarrow_{\$} \mathsf{Sign}(\mathsf{sk}, m)$ that $\mathsf{Verify}(\mathsf{vk}, m, s) = 1$.

Definition A.4 (Strong one-time signatures) *Signature scheme* SIG *is (τ, ε)-strong existential unforgeable under one-time chosen-message attacks (*S-OTS*) iff*

$$\Pr[\mathsf{Exp}^{\mathsf{S\text{-}OTS}}_{\mathsf{SIG},\mathcal{F}}(n) = 1] \leq \varepsilon$$

holds for any PPT adversary $\mathcal{F}$ with running time τ, where $\mathsf{Exp}^{\mathsf{S\text{-}OTS}}_{\mathsf{SIG},\mathcal{F},q}(n)$ is defined in Table 3.

Experiment $\mathsf{Exp}^{\mathsf{S\text{-}OTS}}_{\mathsf{SIG},\mathcal{F}}(n)$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(1^n)$;
$(m^*, s^*) \leftarrow_{\$} \mathcal{F}^{\mathtt{OSign}(\cdot)}(\mathsf{vk})$, where the oracle
$\mathtt{OSign}(\cdot) := \mathsf{Sign}(\mathsf{sk}, \cdot)$ can be asked at most once
If $\mathsf{Verify}(\mathsf{vk}, m^*, s^*) = 1$ and $(m^*, s^*) \neq (m_1, s_1)$
then return 1, else return 0.

Table 3: Security experiment for strong one-time signature.

UC framework We focus in this paper on protocols whose security is proven in the universal composability framework, more precisely on password-authenticated key-exchange (PAKE) and commitments. In a nutshell, this framework allows the protocols to remain secure when composed in an arbitrary environment, which reveals particularly useful in the case of password-based protocols. This simulation-based notion of security relies on the use of ideal functionalities, which capture all the necessary properties of the protocols and the means of an adversary. For clarity, we defer in Appendix C the ideal functionalities of PAKE and commitments. The interested reader is referred to [15, 18, 39] for details.

Smooth Projective Hash Functions Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup [21] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) *hashing* key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has found applications in various contexts in cryptography (*e.g.* [28, 34, 2]), and we will rely on it, for most of our constructions.

Definition A.5 (Smooth Projective Hashing System) *A Smooth Projective Hash Function over a language $\mathcal{L} \subset X$ onto a set $\mathcal{H}$, is defined by five algorithms* $(\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$*:*

- $\mathsf{Setup}(1^n)$ *where 1^n is the security parameter, generates the global parameters* params *of the scheme, and the description of an $\mathcal{NP}$ language $\mathcal{L}$;*
- $\mathsf{HashKG}(\mathcal{L}, \mathsf{params})$*, outputs a hashing key* hk *for the language $\mathcal{L}$;*
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{params}), W)$*, derives the projection key* hp*, possibly depending on the word W [28, 2] thanks to the hashing key* hk*.*
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{params}), W)$*, outputs a hash value $H \in \mathcal{H}$, thanks to the hashing key* hk*, and W*
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{params}), W, w)$*, outputs the hash value $H' \in \mathcal{H}$, thanks to the projection key* hp *and the witness w that $W \in \mathcal{L}$.*

In the following, we consider $\mathcal{L}$ as a hard-partitioned subset of X, i.e. it is computationally hard to distinguish a random element in $\mathcal{L}$ from a random element in $X \setminus \mathcal{L}$.

A Smooth Projective Hash Function should satisfy the following properties:

- *Correctness*: Let $W \in \mathcal{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{params}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{params}), W, w)$.
- *Smoothness*: For all $W \in X \setminus \mathcal{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathcal{L}, \mathsf{params}, W, \mathsf{hp}, H) \;\middle|\; \begin{array}{l} \mathsf{params} = \mathsf{Setup}(1^n), \mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{params}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{params}), W), H = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{params}), W) \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathcal{L}, \mathsf{params}, W, \mathsf{hp}, H) \;\middle|\; \begin{array}{l} \mathsf{params} = \mathsf{Setup}(1^n), \mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{params}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{params}), W), H \xleftarrow{\$} \mathcal{H} \end{array} \right\}.$$

- *Pseudo-Randomness*: If $W \in \mathcal{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable: for any adversary $\mathcal{A}$ within reasonable time

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(n) = \Pr_{\Delta_1}[\mathcal{A}(\mathcal{L}, \mathsf{params}, W, \mathsf{hp}, H) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathcal{L}, \mathsf{params}, W, \mathsf{hp}, H) = 1] \text{ is negligible.}$$

In the special case of Lattices, Katz and Vaikuntanathan have proposed a relaxed definition of Smooth Projective Hash Functions with only an approximated Correctness. This notion echoes to the approximate decryption on ciphertext, where in some cases because of the noise, a decryption on the message might not lead to the initial ciphertext, and so here, in some negligible occasions, an honestly computed projective hash for a word in the language may not be equal to the corresponding hash.

A.2 Useful Tools

A.2.1 Trapdoor Commitment

We construct a trapdoor commitment from SIS assumption, following the chameleon hash from [20] but with the Micciancio-Peikert trapdoor generation [41] (see Lemma A.14 for more details). A chameleon hash is equivalent to the trapdoor commitment, if one view the perfect uniformity of the chameleon hash as the perfect hiding of the commitment, the collision-resistance as the computational binding, and the chameleon property as the perfect trapdoor opening.

For simplicity, we only present the scheme and the security proof is straightforward following the proof of Lemma 4.1 in [20]:

Let $k = \lceil \log q \rceil = O(\log n)$ 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 $s = O(\sqrt{nk})$ be a Gaussian parameter. Define the randomness space $\mathcal{R} := D_{\mathbb{Z}^m,s\cdot\omega(\sqrt{\log n})}$. Then the commitment scheme $\mathsf{COM}_{\mathsf{SIS}} = (\mathsf{CKG}, \mathsf{Commit}, \mathsf{Open})$ with message space $\{0,1\}^\ell$ is defined as follows:

- $\mathsf{CKG}(1^n)$: choose a random matrix $\mathbf{A}_0 \leftarrow_\$ \mathbb{Z}_q^{n\times\ell}$. Sample $(\mathbf{A}_1, \mathbf{R}_1) \leftarrow_\$ \mathsf{GenTrap}^{\mathcal{D}}(1^n, 1^m, q)$. Define $\mathsf{ck} := (\mathbf{A}_0, \mathbf{A}_1)$ and $\mathsf{td} := \mathbf{R}_1$.
- $\mathsf{Commit}(\mathsf{ck}, \mathbf{m})$: choose a vector $\mathbf{r}$ from the Gaussian distribution $D_{\mathbb{Z}^m,s\cdot\omega(\sqrt{\log n})}$, $\mathbf{r} \leftarrow D_{\mathbb{Z}^m,s\cdot\omega(\sqrt{\log n})}$. Compute the commitment value $\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{A}_1\mathbf{r}$. Return the commitment $\mathbf{c}$ and the opening information $\mathbf{r}$.
- $\mathsf{Open}(\mathsf{ck}, \mathbf{m}, \mathbf{c}, \mathbf{r})$: accept if $\|\mathbf{r}\| \le s\cdot\omega(\sqrt{\log n})\cdot\sqrt{m}$ and $\mathbf{c} = \mathbf{A}_0\mathbf{m} + \mathbf{A}_1\mathbf{r}$; otherwise, reject.

As a trapdoor commitment, there exist an efficient trapdoor opening algorithm for $\mathsf{COM}_{\mathsf{SIS}}$:

- $\mathsf{Topen}(\mathsf{td}, (\mathbf{m}_0, \mathbf{r}_0), \mathbf{m}_1)$: 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 $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_1),s\cdot\omega(\sqrt{\log n})}$, $\mathbf{r}_1 \leftarrow_\$ \mathsf{SampleD}(\mathbf{R}_1, \mathbf{A}_1, \mathbf{u}, s)$.

Note that the trapdoor opening Topen works in a stronger sense, where Topen inputs the commitment value $\mathbf{c}_0$ instead of $(\mathbf{m}_0, \mathbf{r}_0)$, since in the construction $\mathbf{A}_0\mathbf{m}_0 + \mathbf{A}_1\mathbf{r}_0 = \mathbf{c}_0$.

A.2.2 Strong One-Time Signature

The simplest way to obtain the strong one-time signature from lattices is to implement Lamport's construction with the following SIS function $f_{\mathbf{A}} : \mathbf{x} \in \{0,1\}^m \mapsto \mathbf{A}\mathbf{x} \bmod q$ for appropriate parameter choices of $n, q, m \ge n\log q$ and a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. It is easy to see this Lamport construction is a strong one-time signature, since $f_{\mathbf{A}}$ is not only one-way but also collision resistant.

An alternative efficient construction was proposed by Lyubashevsky and Micciancio [40], which shows that a ring-variant of SIS leads to a strong one-time signature scheme with quasi-linear efficiency.

To be consistent with the tag-based encryption from Sect. 3.2.1, we use the following strong one-time signature from our trapdoor commitment. We firstly present the generic construction from trapdoor commitments and its security proof and then implement it by using the scheme from Sect. A.2.1.

GENERIC SCHEME. Let $\mathsf{COM} = (\mathsf{CKG}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Topen})$ be a trapdoor commitment. Then our strong one-time signature $\mathsf{SIG}_{\mathsf{Trap}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is defined as follows. Without loss of generality, we assume the message space is the same as the commitment space; otherwise, one can use a collision-resistant hash function to make them consistent.

- $\mathsf{Gen}(1^n)$: sample $(\mathsf{ck}_1, \mathsf{td}_1) \leftarrow_\$ \mathsf{CKG}(1^n)$ and $(\mathsf{ck}_2, \mathsf{td}_2) \leftarrow_\$ \mathsf{CKG}(1^n)$. Compute $(\hat{c}, \hat{r}) \leftarrow_\$ \mathsf{Commit}(\mathsf{ck}_1, 0)$ where 0 can be any dummy message. Define $\mathsf{vk} := (\mathsf{ck}_1, \mathsf{ck}_2, \hat{c})$ and $\mathsf{sk} := (\mathsf{td}_1, \hat{r})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: compute $(c_2, r_2) \leftarrow_\$ \mathsf{Commit}(\mathsf{ck}_2, m)$ and trapdoor open $r_1 \leftarrow_\$ \mathsf{Topen}(\mathsf{td}_1, (0, \hat{r}), c_2)$. Define the signature as $s := (r_1, c_2, r_2)$.
- $\mathsf{Verify}(\mathsf{vk}, m, s = (r_1, c_2, r_2))$: check if $\mathsf{Open}(\mathsf{ck}_2, m, c_2, r_2) = 1$ and $\mathsf{Open}(\mathsf{ck}_1, c_2, \hat{c}, r_1) = 1$.

Correctness is easy to verify by the perfect trapdoor opening of COM.

Theorem A.6 (Security of $\mathsf{SIG}_{\mathsf{Trap}}$) *If COM is a trapdoor commitment with (τ, ε)-binding, then $\mathsf{SIG}_{\mathsf{Trap}}$ is a (τ', ε')-secure strong one-time signature with $\varepsilon' \le 2\varepsilon$ and $\tau' = O(\tau)$.*

We postpone the proof of this theorem in the Appendix D.

IMPLEMENTATION FROM SIS. We fix the parameter n, p, k, m and also the Gaussian parameter s and Gaussian distribution $\mathcal{D}$ in the same way as Sect. A.2.1. Let $\mathfrak{H}_K : \mathbb{Z}_q^n \mapsto \{0,1\}^\ell$ be a collision-resistant hash function, which is easy to construction. A naive way to implement $\mathfrak{H}_K$ is to view the element $\mathbf{x} \in \mathbb{Z}_q^n$ as a ℓ-bit string for an appropriate ℓ. Then the strong one-time signature $\mathsf{SIG}_{\mathsf{SIS}}$ from SIS is defined as follows:

- $\mathsf{Gen}(1^n)$: pick a random matrix $\mathbf{A}_0 \leftarrow_{\$} \mathbb{Z}_q^{n\times\ell}$ and sample $(\mathbf{A}_1, \mathbf{R}_1) \leftarrow_{\$} \mathsf{GenTrap}^{\mathcal{D}}(1^n, 1^m, q)$ and $(\mathbf{A}_2, \mathbf{R}_2) \leftarrow_{\$} \mathsf{GenTrap}^{\mathcal{D}}(1^n, 1^m, q)$. Choose a uniformly random vector $\mathbf{u} \leftarrow_{\$} \mathbb{Z}_q^n$. Define the verification key $\mathsf{vk} := (\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{u}, \mathfrak{H}_K)$ and the signing key $\mathsf{sk} := \mathbf{R}_1$.
- $\mathsf{Sign}(\mathsf{sk}, \mathbf{m})$: sample $\mathbf{r}_2 \leftarrow D_{\mathbb{Z}^m, s\cdot\omega(\sqrt{\log n})}$. Compute $\mathbf{c}_2 = \mathbf{A}_0\mathbf{m} + \mathbf{A}_2\mathbf{r}_2$ and $\mathbf{y} = \mathbf{u} - \mathbf{A}_0 \cdot \mathfrak{H}_K(\mathbf{c}_2)$. Sample $\mathbf{r}_1 \leftarrow \mathsf{SampleD}(\mathbf{R}_1, \mathbf{A}_1, \mathbf{y}, s)$. Define the signature $\mathbf{s} := (\mathbf{r}_1, \mathbf{c}_2, \mathbf{r}_2)$.
- $\mathsf{Verify}(\mathsf{vk}, (\mathbf{m}, \mathbf{s}))$: phase $\mathbf{s}$ as $\mathbf{s} = (\mathbf{r}_1, \mathbf{c}_2, \mathbf{r}_2)$. Check if $\|\mathbf{r}_1\| \le s \cdot \omega(\sqrt{\log n}) \cdot \sqrt{m}$ and $\|\mathbf{r}_2\| \le s \cdot \omega(\sqrt{\log n}) \cdot \sqrt{m}$. If both hold, then continue to check if $\mathbf{c}_2 = \mathbf{A}_0\mathbf{m} + \mathbf{A}_2\mathbf{r}_2$ and $\mathbf{u} = \mathbf{A}_0 \cdot \mathfrak{H}_K(\mathbf{c}_2) + \mathbf{A}_1\mathbf{r}_1$.

A.3 Facts about Lattices

For integers n, m and for a prime q, let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. The m-dimensional integer lattice $\Lambda^{\perp}(\mathbf{A})$ is defined as

$$\Lambda^{\perp}(\mathbf{A}) := \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \bmod q\}.$$

For any $\mathbf{u} \in \mathbb{Z}_q^n$, define the coset

$$\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}) := \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{u} \bmod q\}.$$

Leftover Hash Lemma A distribution D is said to be ε-uniform if its statistical distance from the uniform distribution is at most ε. Let X and Y be finite sets. A family $\mathcal{H}$ of hash functions from X to Y is said to be *pairwise-independent* if for all distinct $x, x' \in X$, $\Pr_{h\leftarrow H}[h(x) = h(x')] = 1/|Y|$.

Lemma A.7 (Leftover Hash Lemma [33]) *Let $\mathcal{H}$ be a family of pairwise-independent hash functions from X to Y. Suppose that $h \leftarrow \mathcal{H}$ and $x \leftarrow X$ are chosen uniformly and independently. Then, $(h, h(x))$ is $\frac{1}{2}\sqrt{|Y|/|X|}$-uniform over $\mathcal{H} \times Y$.*

For our applications, note that if $\mathbf{A} \in \mathbb{Z}^{m\times n}$ is chosen uniformly at random, and if $m \ge \Theta(n \log q)$ then the functions $f_{\mathbf{A}} : \mathbf{x} \in \{0,1\}^m \mapsto \mathbf{A} \cdot \mathbf{x} \in \mathbb{Z}^n$ form a pairwise-independent family, in particular for $\mathbf{x} \leftarrow \mathcal{U}(\{0,1\})$, $\mathbf{A}\mathbf{x}$ is negligibly close to the uniform distribution $\mathcal{U}(\mathbb{Z}_q^n)$.

A.3.1 Discrete Gaussian Distributions

Definition A.8 *The (unnormalized) weight of Gaussian distribution of parameter $s \in \mathbb{R}$ and center $c \in \mathbb{R}$ at $x \in \mathbb{R}$ is defined by $\rho_{s,c}(x) = \exp\big(-\pi\frac{(x-c)^2}{s^2}\big)$, and more generally, the spherical Gaussian distribution of parameter $s > 0$ and center $\mathbf{c} \in \mathbb{R}^n$ is defined over $\mathbb{R}^n$ as*

$$\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi\frac{(\mathbf{x}-\mathbf{c})^2}{s^2}}.$$

The discrete Gaussian distribution over $\mathbb{Z}$ is defined by the probabilities $D_{\mathbb{Z},s,c}(x) = \rho_{s,c}(x)/\rho_{s,c}(\mathbb{Z})$ for any $x \in \mathbb{Z}$, and more generally, over a lattice L by

$$D_{L,s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/\rho_{s,\mathbf{c}}(L) \text{ for any } \mathbf{x} \in L.$$

The following lemmas are useful for this paper:

Definition A.9 [Smoothing Parameter [42]] For any n-dimensional lattice L and any real $\epsilon > 0$, the *smoothing parameter* $\eta_\epsilon(L)$ (see [42]) is the smallest real $s > 0$ such that $\rho_{1/s}(\hat{L} \setminus \{\mathbf{0}\}) \le \epsilon$.

Lemma A.10 (Implicit in [42, Lemma 4.4]) *For any integer full rank matrix $\mathbf{B} \in \mathbb{Z}^{n\times m}$ and $\epsilon \in (0,1)$, if $\sigma \ge \eta_\epsilon(\Lambda^{\perp}(\mathbf{B}))$ we have that the distribution of $\mathbf{w} = \mathbf{B} \cdot \mathbf{v} \bmod q$ for $\mathbf{v} \leftarrow D_{\mathbb{Z}^m,s}$ is at distance $O(\epsilon)$ from uniform over $\mathbb{Z}_q^n$.*

Lemma A.11 (Lemma 3.3 of [42]) *For any lattice $L \subset \mathbb{R}^m$ of dimension n and any $\epsilon \in (0,1]$,*

$$\eta_\epsilon(L) \le \sqrt{\ln(2n(1+1/\epsilon))/\pi} \cdot \lambda_n(L)$$

where $\lambda_n(L)$ denotes the largest minima of L. In particular, if $\mathbf{B}$ is a basis of L, for any super-logarithmic function $\omega(\log n)$ there exists a negligible function $\epsilon(n)$ such that $\eta_\epsilon(L) \le \|\mathbf{B}\| \sqrt{\omega(\log n)}$.

Corollary A.12 (Smoothing Lemma over $\mathbb{Z}$) *The statistical distance between $D_{\mathbb{Z},d\cdot\omega(\sqrt{\log\lambda})} \bmod d$ the discrete Gaussian taken modulo $d \in \mathbb{Z}$, and $\mathcal{U}(\mathbb{Z}_d)$ the uniform distribution over $\mathbb{Z}_d$, is negligible in λ.*

Lemma A.13 (Smoothing Lemma for randome lattice, Corollary 5.4 of [30]) *Let n be a positive integer and q be a prime, and let integer $m \ge 2n\log q$. Then for all but a $2q^{-n}$ fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and for any $s \ge \omega(\sqrt{\log m})$, the distribution of the syndrome $\mathbf{y} = \mathbf{A}\mathbf{x} \mod q$ is statically close to uniform over $\mathbb{Z}_q^n$, where $\mathbf{x}$ is from $D_{\mathbb{Z}^m,s}$.*

Lemma A.14 (Theorem 5.1 of [41]) *There is an efficient randomized algorithm* $\mathsf{GenTrap}^{\mathcal{D}}(1^n, 1^m, q)$ *that, given any integers* $n \geq 1$, $q \geq 2$, *and sufficiently large* $m = O(n \log q)$, *outputs a parity-check matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and a trapdoor* $\mathbf{R}$ *such that the distribution of* $\mathbf{A}$ *is* $\mathsf{negl}(n)$*-far from uniform and* $\mathbf{R}$ *is sampled from the Gaussian* $\mathcal{D}$. *Moreover, there are efficient algorithms* Invert *and* $\mathsf{SampleD}$ *that with overwhelming probability over all random choices, do the followings:*

- *For* $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$ *is arbitrary and either* $\|\mathbf{e}\| < q/O(\sqrt{n \log q})$ *or* $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, \alpha q}$ *for* $1/\alpha \geq \sqrt{n \log q} \cdot \omega(\sqrt{\log q})$, *the deterministic algorithm* $\mathsf{Invert}(\mathbf{R}, \mathbf{A}, \mathbf{b})$ *outputs* $\mathbf{s}$ *and* $\mathbf{e}$*;*
- *For any* $\mathbf{u} \in \mathbb{Z}_q^n$ *and large enough* $s = O(\sqrt{n \log q})$, *there is an efficient randomized algorithm* $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$ *that samples from a distribution with* $\mathsf{negl}(n)$ *statistical distance of* $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}), s \cdot \omega(\sqrt{\log n})}$.

B The Double Micciancio Peikert Encryption

Recently, Lindell [39] proposed a highly efficient UC commitment, we are going to use an adaptation that does not need to be UC secure. We will then show that the decommitment check can be done in an implicit way with an appropriate smooth projective hash function. Basically, the technique consists in encrypting b in $\mathcal{C} = \mathsf{MP}(\mathfrak{L}, b; \mathbf{r}, \mathsf{vk})$, also getting $\mathbf{T} = \mathfrak{H}_K(\mathfrak{L}, \mathsf{vk})$, and then encrypting 0 in $\mathcal{C}' = \mathsf{MP}^*(\mathfrak{L}, 0, \mathbf{T}; \mathbf{s})$, with the same $\mathbf{T}$. For a given challenge ζ, we can see that $\forall i, \mathcal{C} + \zeta_i \mathcal{C}'_i = \mathsf{MP}^*(\mathfrak{L}, b, \mathbf{T}; \mathbf{r} + \zeta_i \mathbf{s}_i)$, this linear redundancy of control is here to exponentially reduce the adversary chance of passing an invalid ciphertext.

This section may be seen as redundant with the presentation in Section 5, it is just here to clarify some techniques, and underlines the fact that we do not need this commitment to be UC, to use it in our UC PAKE, only a sketch of the proof is given as it is more meticulously handled in the UC part in section D.4.

It makes use of an equivocable Trapdoor commitment as described in Section A.2.1.

To achieve the CCA-2 security each MP ciphertext is sent with a Strong-OTS, which is verified immediately by the verifier, for clarity they are omitted in the scheme below, but we insist that the verification key of the signature is included in the computation of the tag (even if we omit it from now on for simplicity).

We define $\mathfrak{n} = O(n)$.

B.1 Description.

Our n-bits vector commitment, which includes labels, is depicted on Figure 8. We assume we commit vectors of bits, we will note $\mathbf{M} = \mathbf{ECC}(\mathbf{b})$ for sake of simplicity.

Note that for this commitment scheme, we use one vector ζ shared for all bits, for the version with SPHF implicit verification, if we study languages more complex that equality, one can have to use independent components, for sake of simplicity we note $\mathcal{C}'$ the set of $\mathfrak{n}\mathcal{C}'_i$ and consider operation on it as if it was only one ciphertext.

- Setup: Generates the Micciancio Peikert encryption key ek, a collision-resistant hash function $\mathfrak{H}_K$ mapping to the tagspace, and the key for a trapdoor commitment ck;
- $\mathsf{Commit}(\mathfrak{L}, \mathbf{M}; \mathbf{r}, \mathbf{s_i}, \mathbf{e_r}, \mathbf{e_{s_i}}, \mathbf{t})$:

$$
\begin{array}{lcl}
(\mathcal{C}, \mathcal{C}') \leftarrow n - \mathsf{DMP}(\mathfrak{L}, \mathsf{ek}, \mathbf{b}, \mathbf{0}; \mathbf{r}, \mathbf{s}, \mathbf{T} = \mathfrak{H}_K(\mathfrak{L}), \mathbf{e_r}, \mathbf{e_s}) & & \\
\chi = \mathfrak{H}_K(\mathbf{b}, \mathcal{C}') & & \\
\mathcal{C}'' = \mathsf{Trap.Commit}(\chi; \mathbf{t}) & \xrightarrow{\mathcal{C}, \mathcal{C}''} & \\
\zeta \overset{?}{\neq} 0 & \xleftarrow{\zeta} & \zeta \leftarrow D^{\mathfrak{n}}_{\mathbb{Z}, \omega(\sqrt{\log \lambda})} \\
\mathbf{z} = (\mathbf{r} + \zeta \cdot \mathbf{s}), \mathbf{e_z} = (\mathbf{e_r} + \zeta \cdot \mathbf{e_s}) & & \\
\text{Erase } (\mathbf{r}, \mathbf{s}, \mathbf{e_r}, \mathbf{e_s}) & &
\end{array}
$$

- $\mathsf{Decommit}(\mathfrak{L}, \mathcal{C}, \mathcal{C}', \zeta)$:

$$
\begin{array}{lcl}
 & \xrightarrow{\mathcal{C}', \mathbf{T}, \mathbf{M}, \mathbf{z}, \mathbf{e_z}} & \text{compute } \mathbf{T} \text{ from } \mathfrak{L}, \mathsf{vk} \\
 & & \chi = \mathfrak{H}_K(\mathbf{M}, \mathcal{C}'), \\
 & & \mathsf{Trap.Open}(\mathsf{ck}, \mathcal{C}'', \mathfrak{H}_K(\mathcal{C}'), \mathbf{t})? \\
 & & \mathcal{C} + \zeta \mathcal{C}' = n - \mathsf{MP}^*(\mathfrak{L}, \mathbf{M}, \mathbf{T}; \mathbf{z}, \mathbf{e_z})
\end{array}
$$

Figure 8: $n - \mathsf{DMP}$ Commitment Scheme

B.2 Analysis.

Let us briefly show the properties of this commitment:

- Hiding property: $\mathbf{M}$ is committed in the Trapdoor commitment $\mathcal{C}''$, that does not leak any information, and in the $n-\mathsf{MP}$ encryption $\mathcal{C}$, that is indistinguishable, even with access to the decryption oracle (extractability). This also implies non-malleability when we add the Strong One-Time Signature.
- Binding property: $\mathbf{M}$, after having been hashed, is committed in the Trapdoor commitment $\mathcal{C}''$, that is computationally binding.
- Extractability: using the decryption key of the MP encryption scheme, one can extract $\mathbf{b}$ from $\mathcal{C}$. Later, one has to open the ciphertext $\mathcal{C}+\zeta\mathcal{C}'$ with $\mathbf{b}'$. If $\mathcal{C}'$ was generated honestly this value is equal to $\mathbf{b}$ with overwhelming probability.

  If all the ciphertext in $\mathcal{C}'$ are not an honestly generated encryption of 0, then the value $\mathbf{b}'$ could not be predicted by the adversary at the beginning of the protocol. As this value is in $\{0,1\}$, we have a good prognostic for a value with probability 0.5, as we have several $\mathfrak{n}=O(n)$ challenges ζ_i, so that probability that all of them match to the same value, and that this value was predicted falls in $O(2^{-\mathfrak{n}})$.
- Equivocability: if one wants to open with $\mathbf{M}'$, one can compute $\mathbf{N}=(\mathbf{M}'-\mathbf{M})/\zeta$, encrypt $\mathbf{N}$ in $\mathcal{C}'=n-\mathsf{MP}^*(\mathfrak{L},\mathbf{N},\mathbf{T};\mathbf{s})$, and update χ and $\mathbf{t}$, using the trapdoor for equivocability.

To allow an implicit verification with SPHF, one omits to send $\mathbf{M}$ and $\mathbf{z},\mathbf{e}_z$, but make an implicit proof of their existence. Therefore, $\mathbf{M}$ cannot be committed/verified in $\mathcal{C}''$, which has an impact on the binding property: $\mathcal{C}$ and $\mathcal{C}''$ are not bound to a specific $\mathbf{M}$, even in a computational way. However, as said above, if $\mathcal{C}''$ contains a ciphertext $\mathcal{C}'$ non honestly generated on 0, the actual committed value will depend on ζ: has its i-component, where N_i, uniformly distributed, when ζ is uniformly distributed in $\{0,1\}$, the probability that given a dishonest $\mathcal{C}'_i$, and $\mathfrak{n}$ challenges all the $\mathcal{C}''_i$ decommits to the same $\bar{b}$ is close to $O(2^{-\mathfrak{n}})$, hence negligible.

B.3 Security of the Encryption Scheme

We are going to sketch the security of the DMP encryption. This is mostly a transposition of the Double Cramer Shoup from [8], no particularity arises during the proof. (As we need to add a Strong-OTS to the Micciancio Peikert like encryption, the security of this signature will also appear).

We described everything for the 1 bit scheme.

B.3.1 Security model.

This scheme is indistinguishable against *partial-decryption chosen-ciphertext* attacks, where a partial-decryption oracle only is available, but even when we allow the adversary to choose M and N in two different steps (see the security game below), under the LWE assumption and if one uses a collision-resistant hash function $\mathfrak{H}_K$ and a Strong-OTS.

Indistinguishability against partial-decryption chosen-ciphertext attacks, in two steps: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls $\mathtt{FIND}_M$, $\mathtt{FIND}_N$ and $\mathtt{GUESS}$. In the first stage $\mathtt{FIND}_M$, it receives the encryption key ek; in the second stage $\mathtt{FIND}_N$, it receives the encryption of M_b: $\mathcal{C}^*=\mathsf{Encrypt}(\mathfrak{L},\mathsf{ek},M_b)$; in the last stage $\mathtt{GUESS}$ it receives the encryption of N_b: $\mathcal{C}'^*=\mathsf{Encrypt}'(\mathfrak{L},\mathsf{ek},\mathbf{T}^*,N_b)$, where $\mathbf{T}^*$ is the value involved in $\mathcal{C}$. During all these stages, it can make use of the oracle $\mathsf{ODecrypt}(\mathfrak{L},\mathcal{C})$, that outputs the decryption of $\mathcal{C}$ under the label $\mathfrak{L}$ and the challenge decryption key dk, using $\mathsf{PDecrypt}(\mathfrak{L},\mathsf{dk},\mathcal{C})$. The input queries $(\mathfrak{L},\mathcal{C})$ are added to the list $\mathcal{CT}$.

$\mathsf{Exp}^{\mathsf{ind\text{-}pd\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(n)$
1. $\mathsf{params}\leftarrow\mathsf{Setup}(1^n)$; $(\mathsf{ek},\mathsf{dk})\leftarrow\mathsf{Gen}(\mathsf{params})$
2. $(\mathfrak{L}^*,M_0,M_1)\leftarrow\mathcal{A}(\mathtt{FIND}_M:\mathsf{ek},\mathsf{ODecrypt}(\cdot,\cdot))$
3. $\mathcal{C}^*\leftarrow\mathsf{Encrypt}(\mathfrak{L}^*,\mathsf{ek},M_b)$
4. $(N_0,N_1)\leftarrow\mathcal{A}(\mathtt{FIND}_N:\mathcal{C}^*,\mathsf{ODecrypt}(\cdot,\cdot))$
5. $\mathcal{C}'^*\leftarrow\mathsf{Encrypt}'(\mathfrak{L}^*,\mathsf{ek},\mathbf{T}^*,N_b)$
6. $b'\leftarrow\mathcal{A}(\mathtt{GUESS}:\mathcal{C}'^*,\mathsf{ODecrypt}(\cdot,\cdot))$
7. $\mathtt{IF}\ (\mathfrak{L}^*,\mathcal{C}^*)\in\mathcal{CT}\ \mathtt{RETURN}\ 0$
8. $\mathtt{ELSE\ RETURN}\ b'$

The advantages are, where q_d is the number of decryption queries:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{\mathcal{E}}(\mathcal{A}) &= \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}pd\text{-}cca}-1}_{\mathcal{E},\mathcal{A}}(n)=1]-\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}pd\text{-}cca}-0}_{\mathcal{E},\mathcal{A}}(n)=1]\\ \mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{\mathcal{E}}(q_d,t) &= \max_{\mathcal{A}\le t}\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{\mathcal{E}}(\mathcal{A}).\end{aligned}$$

Theorem B.1 *The DMP encryption scheme is IND-PD-CCA if $\mathfrak{H}_K$ is a collision-resistant hash function family, under the* LWE *and* SIS *assumption.*

Corollary B.2 *The Multiple $n - \mathsf{MP}$ encryption scheme is IND-CCA if $\mathfrak{H}_K$ is a collision-resistant hash function family, under the* LWE *and* SIS *assumption.*

B.3.2 Security proof. (sketch)

In this section, we only give a high level idea of what happens in the ind-pd-cca security proof, for that we show how to use $\mathcal{A}$ to answer to a CCA-2 challenger.

- In the initial game $\mathcal{G}_0$,
  - A's decryption queries are answered by $\mathcal{B}$, simply using the normal decryption.
  - When $\mathcal{A}$ submits the first challenge M_0, M_1, with a label $\mathfrak{L}^*$, $\mathcal{B}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$ and encrypts $\mathbf{M}_b$:
  - When $\mathcal{A}$ submits the second challenge N_0, N_1, and do a MP^* encryption of N_b (i.e. without the Strong-OTS).
  - When $\mathcal{A}$ returns b', $\mathcal{B}$ outputs $b' \stackrel{?}{=} b$.
- In game $\mathcal{G}_1$, we handle cases where $M_0 \neq M_1$, we simply rely on the simulator of the CCA-2 security of the MP encryption (with the Strong-OTS):
  - Decryption queries are answered by the simulator of the CCA-2.
  - When $\mathcal{A}$ submits the first challenge M_0, M_1, with a label $\mathfrak{L}^*$, the simulator from CCA-2 gives us a challenge C_b
  - When $\mathcal{A}$ submits the second challenge N_0, N_1:
    - If $N_0 = N_1$, $\mathcal{B}$ proceeds honestly
    - If $M_0 = N_0$, and so $M_1 = N_1$, one simple randomizes C_b (without the signature it is easy), to generate C'_b
    - Else this means $N_b = M_b + 1 \bmod 2$, hence we randomize into C'_b by starting from $C_b - (\mathbf{0}|\mathsf{ECC}(1))$.
  - When $\mathcal{A}$ returns b', $\mathcal{B}$ forwards b' to the CCA-2 challenger.
- In game $\mathcal{G}_2$, we handle $M_0 = M_1$, we will once again use a CCA-2 challenger, but with $N_0, N_1, \mathfrak{L}^*$, as we can do the first encryption honestly.

As $\mathcal{A}$ answers the correct bit b' with non-negligible probability and our simulation is perfect, we manage to break the CCA-2 security of the revisited Micciancio Peikert Scheme, hence either the LWE and SIS assumptions, or the collision resistance of the Hash Function.

C UC Framework and Ideal Functionalities

C.1 Quick Presentation of the UC Framework

Throughout this paper we assume basic familiarity with the universal composability framework. Here we provide a brief overview of the framework. The interested reader is referred to [15] for complete details. In a nutshell, security in the UC framework is defined in terms of an ideal functionality $\mathcal{F}$, which is basically a trusted party that interacts with a set of players to compute some given function f. In particular, the players hand their input to $\mathcal{F}$ which computes f on the received inputs and gives back to each player the appropriate output. Thus, in this idealized setting, security is inherently guaranteed, as any adversary, controlling some of the parties, can only learn (and possibly modify) the data of corrupted players. In order to prove that a candidate protocol π realizes the ideal functionality, one considers an environment $\mathcal{Z}$, which is allowed to provide inputs to all the participants and that aims to distinguish the case where it receives the outputs produced from a real execution of the protocol (involving all the parties and an adversary $\mathcal{A}$, controlling some of the parties and the communication among them), from the case where it receives outputs obtained from an ideal execution of the protocol (involving only dummy parties interacting with $\mathcal{F}$ and an ideal adversary $\mathcal{S}$ also interacting with $\mathcal{F}$). Then we say that π realizes the functionality $\mathcal{F}$ if for every (polynomially bounded) $\mathcal{A}$, there exists a (polynomially bounded) $\mathcal{S}$ such that no (polynomially bounded) $\mathcal{Z}$ can distinguish a real execution of the protocol from an ideal one with a significant advantage. In particular, the universal composability theorem assures us that π continues to behave like the ideal functionality even if it is executed in an arbitrary network environment.

As a consequence, the formal security proof is performed by showing that for any external entity, that gives inputs to the honest players and gets the outputs but that also controls the adversary, the executions in the two above worlds are indistinguishable. More concretely, in order to prove that a protocol $\mathcal{P}$ realizes an ideal functionality $\mathcal{F}$, we consider an environment $\mathcal{Z}$ which can choose inputs given to all the honest players and receives back the outputs they get, but which also controls an adversary $\mathcal{A}$. Its goal is to distinguish in which case it is: either the real world with concrete interactions between the players and the adversary, or the ideal world in which players simply forward everything to and from the ideal functionality and the adversary interacts with a simulator $\mathcal{S}$ to attack the ideal functionality. We have to build a simulator $\mathcal{S}$ that makes the two views indistinguishable to the environment: since the combination of the adversary and the simulator

cannot cause any damage against the ideal functionality, this shows that the adversary cannot cause any damage either against the real protocol.

The main constraint is that the simulator cannot rewind the execution as often done in classical proofs, since it interacts with an adversary under the control of the environment: there is no possible rewind in the real word, it is thus impossible too in the ideal world.

The adversary $\mathcal{A}$ has access to the communication but nothing else, and namely not to the inputs/ouputs for the honest players. In case of corruption, it gets complete access to inputs and the internal memory of the honest player, and then gets control of it.

C.2 The Ideal Functionality of Password-Based Authenticated Key-Exchange

The functionality $\mathcal{F}_{\mathrm{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving a query $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ from party P_i:**

  Send $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to $\mathcal{S}$. If this is the first NewSession query, or if this is the second NewSession query and there is a record $(\mathsf{sid}, \mathsf{ssid}, P_j, P_i, \mathsf{pw}')$, then record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ and mark this record fresh.

- **Upon receiving a query $(\mathsf{TestPwd}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{pw}')$ from the adversary S:**

  If there is a record of the form (P_i, P_j, pw) which is fresh, then do: If $pw = pw'$, mark the record compromised and reply to $\mathcal{S}$ with "correct guess". If $\mathsf{pw} \neq \mathsf{pw}'$, mark the record interrupted and reply with "wrong guess".

- **Upon receiving a query $(\mathsf{NewKey}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{sk})$ from the adversary S:**

  If there is a record of the form $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$, and this is the first NewKey query for P_i, then:

  - If this record is compromised, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk})$ to player P_i.
  - If this record is fresh, and there is a record (P_j, P_i, pw') with $\mathsf{pw}' = \mathsf{pw}$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was fresh at the time, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk}')$ to P_i.
  - In any other case, pick a new random key sk' of length n and send $(\mathsf{sid}, \mathsf{ssid}, sk')$ to P_i.

  Either way, mark the record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ as completed.

Figure 9: Ideal Functionality for PAKE $\mathcal{F}_{\mathbf{pwKE}}$

We present the PAKE ideal functionality $\mathcal{F}_{\mathrm{pwKE}}$ on Figure 9). It was described in [18]. The main idea behind this functionality is as follows: If neither party is corrupted, then they both end up with the same uniformly-distributed session key, and the adversary learns nothing about it (except that it was indeed generated). However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as compromised), then it is granted the right to fully determine its session key. Note that as soon as a party is corrupted, the adversary learns its key: There is in fact nothing lost by allowing it to determine the key. In addition, the players become aware of a failed attempt of the adversary at guessing a password. This is modeled by marking the session as interrupted. In this case, the two players are given independently-chosen random keys. A session that is nor compromised nor interrupted is called fresh. In such a case, the two parties receive the same, uniformly distributed session key. Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password is used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

In case of corruption, the adversary learns the password of the corrupted player, after the NewKey-query, it additionally learns the session key.

C.3 The Ideal Functionality of Commitment

A UC-secure commitment scheme provides all the properties previously given: it should be hiding and binding, but also extractable and equivocable, and even non-malleable.

The ideal functionality is presented on Figure 10. It is borrowed from [15, 39], where a delayed output is an output first sent to the adversary $\mathcal{S}$ that eventually decides if and when the message is actually delivered to the recipient. This models denial of services from the adversary.

$\mathcal{F}_{\mathsf{mcom}}$ with session identifier sid proceeds as follows, running with parties $P_1, \ldots, P_n$, a parameter 1^{1^n}, and an adversary $\mathcal{S}$:

- Commit phase: Upon receiving a message $(\mathsf{Commit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$ from P_i where $x \in \{0,1\}^{\mathsf{polylog}k}$, record the tuple $(\mathsf{ssid}, P_i, P_j, x)$ and generate a delayed output $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to P_j. Ignore further Commit-message with the same $(\mathsf{sid}, \mathsf{ssid})$.
- Reveal phase: Upon receiving a message of the form $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid})$ from party P_i, if a tuple $(\mathsf{ssid}, P_i, P_j, x)$ was previously recorded, then generate a delayed output $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$ to P_j. Ignore further reveal-message with the same $(\mathsf{sid}, \mathsf{ssid})$ from P_i.

Figure 10: Ideal Functionality $\mathcal{F}_{\mathsf{mcom}}$ for Commitment

D Omitted Proofs

D.1 Proof of Theorem 3.2, CCA-1 Security of MP^*

Proof: We prove Theorem 3.2 by defining the following games. Let $\mathcal{M}$ be an adversary against the tbe-stag-cca security (as defined in Def. A.2) of MP^*. Define S_i be the event that $\mathbf{Exp}^{\mathsf{tbe\text{-}stag\text{-}cca}}_{\mathsf{MP}^*,\mathcal{M},Q}(n)$ outputs 1 in game $\mathbf{G}_i$:

Game $\mathbf{G}_0$: This is the original attack game for the tbe-stag-cca security. It is trivial that

$$\Pr[\mathsf{S}_0] = \Pr[\mathbf{Exp}^{\mathsf{tbe\text{-}stag\text{-}cca}}_{\mathsf{MP}^*,\mathcal{M},Q}(n) = 1].$$

Game $\mathbf{G}_1$: In this game, we change the key generation and the way to answer challenge ciphertext. Note that, in the definition of tbe-stag-cca security, we choose a uniform tag $\mathbf{T}^* \leftarrow \mathcal{U}(\mathcal{T})$ related to the challenge ciphertext before the key generation. We change the simulation as follows:

- Key generation: choose $\bar{\mathbf{A}}$ and $\mathbf{R}$ as in the definition. Then define the encryption key to be $\mathsf{ek}' := \mathbf{A}' = [\bar{\mathbf{A}}| - \mathbf{T}^*\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$ and the decryption key $\mathsf{dk} = (\mathbf{R}, \mathbf{A}')$. Note that $[\bar{\mathbf{A}}| - \bar{\mathbf{A}}\mathbf{R}]$ is statistically closed to the uniform distribution over $\mathbb{Z}_q^{n\times m}$, by the Leftover-Hash Lemma A.7. That implies $[\bar{\mathbf{A}}| - \mathbf{T}^*\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$ is also statistically closed to the uniform distribution and independent to $\mathbf{T}^*$.
- Answering decryption queries: on input $(\mathbf{T}, \mathbf{c})$, check if $\mathbf{T} \in \mathcal{T}$. Note that $\mathbf{T}^*$ is still uniform in $\mathcal{T}$ from the adversary point of view, therefore, unless with negligible probability $\mathbf{T} = \mathbf{T}^*$, since $\mathcal{T}$ is exponentially large. Call $(\mathbf{s}_0, \mathbf{e}_0) = \mathsf{Invert}(\mathbf{R}, [\bar{\mathbf{A}}| - \bar{\mathbf{A}}\mathbf{R}], \mathbf{T} - \mathbf{T}^*, \mathbf{c})$ and $(\mathbf{s}_1, \mathbf{e}_1) = \mathsf{Invert}(\mathbf{R}, [\bar{\mathbf{A}}| - \bar{\mathbf{A}}\mathbf{R}], \mathbf{T} - \mathbf{T}^*, \mathbf{c} - (\mathbf{0}, \mathsf{ECC}(1)))$. And we apply Step 2 to 4 as in the original scheme. Since $\mathbf{T} - \mathbf{T}^*$ is invertible, by the correctness of Invert, we can decrypt the ciphertext correctly.
- Computing challenge ciphertext: choose $\mathbf{s}^* \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, $\bar{\mathbf{e}}^* \leftarrow D_{\mathbb{Z},s}^{\bar{m}}$ and $\hat{\mathbf{e}}^* \leftarrow D_{\mathbb{Z},s}^{nk}$ and $\mathbf{e}^* = (\bar{\mathbf{e}^*}, \hat{\mathbf{e}^*})$. Compute $\mathbf{c}_0^* = \bar{\mathbf{A}}^\top \cdot \mathbf{s} + \bar{\mathbf{e}}^\top \in \mathbb{Z}_q^{\bar{m}}$ and $\mathbf{c}_1^* = -\mathbf{c}_0^*\mathbf{R} + \hat{\mathbf{e}}^\top + (\mathbf{0}_{nk-1}, \mathsf{ECC}(M))^\top \in \mathbb{Z}_q^{nk}$. Define $\mathbf{c}^* := (\mathbf{c}_0^*, \mathbf{c}_1^*)$. Note that, by the definition of the simulated encryption key ek', $\mathbf{A} + [\mathbf{0}|\mathbf{T}^*\mathbf{G}] = [\bar{\mathbf{A}}| - \bar{\mathbf{A}}\mathbf{R}]$.

  We show that the distribution of the challenge ciphertext is $\mathsf{negl}(n)$-far from the one of $\mathbf{G}_0$. It is clear that $\mathbf{c}_0^*$ is distributed identically as in $\mathbf{G}_0$. By substitution, we have the following:

$$\begin{aligned}\mathbf{c}_1^* &= -\mathbf{c}_0^*\mathbf{R} + \hat{\mathbf{e}}^\top + (\mathbf{0}_{nk-1}, \mathsf{ECC}(M))^\top \\ &= (-\bar{\mathbf{A}}\mathbf{R})^\top\mathbf{s} + (\bar{\mathbf{e}}^\top\mathbf{R} + \hat{\mathbf{e}}^\top) + (\mathbf{0}_{nk-1}, \mathsf{ECC}(M))^\top\end{aligned}$$

  By the similar argument as in game H_1 of Theorem 6.3, we have $\langle \bar{\mathbf{e}}, \mathbf{r}_i\rangle + \hat{e}_i$ is $\mathsf{negl}(n)$-far from $D_{\mathbb{Z},s}$, where $\mathbf{r}_i$ is the i-th column of $\mathbf{R}$ and $\hat{e}_i$ is the i-th component of $\hat{\mathbf{e}}$. That implies $\bar{\mathbf{e}}^\top\mathbf{R} + \hat{\mathbf{e}}^\top$ is $\mathsf{negl}(n)$-far from $D_{\mathbb{Z},s}^{nk}$.

As shown in above, $\mathbf{G}_1$ and $\mathbf{G}_0$ are statistically closed:

$$|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_0]| \le \mathsf{negl}(n).$$

Game $\mathbf{G}_2$: We only change $\mathbf{c}_0^*$ to be uniformly random in $\mathbb{Z}_q^{\bar{m}}$. We construct the following reduction to show that the difference between $\mathbf{G}_2$ and $\mathbf{G}_1$ is bounded by the decisional LWE assumption dLWE: let $(\bar{\mathbf{A}}, \mathbf{b}) \in \mathbb{Z}_q^{n\times\bar{m}} \times \mathbb{Z}_q^{\bar{m}}$ be the dLWE challenge. We simply simulate the key generation and the decryption oracle as in $\mathbf{G}_1$ except that we set $\mathbf{c}_0^* = \mathbf{b}$ and compute $\mathbf{c}_1^*$ as in $\mathbf{G}_1$. Then if $\mathbf{b}$ follows the LWE distribution, then the challenge ciphertext is identical to that in $\mathbf{G}_1$; otherwise, it is identical to $\mathbf{G}_2$. Thus, we have

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \le \mathsf{AdvdLWE},$$

where AdvdLWE is the advantage of solving dLWE problem.

Again by the Leftover-Hash Lemma, in this game $(\mathbf{c}_0^*, \mathbf{c}_0^*\mathbf{R})$ is $\mathsf{negl}(n)$-far from the uniform distribution over $\mathbb{Z}_q^{\bar{m}} \times \mathbb{Z}_q^{nk}$. Thus, the challenge ciphertext $\mathbf{c}^* = (\mathbf{c}_0^*, \mathbf{c}_1^*)$ is independent from the encrypted message in a statistically sense. Thus, $\Pr[\mathsf{S}_2] = \frac{1}{2} + \mathsf{negl}(n)$, and we have Theorem 3.2. ▌

D.2 Proof of Theorem A.6

We prove Theorem A.6 by defining the following games. Let $\mathcal{F}$ be the forger for $\mathsf{SIG}_{\mathsf{Trap}}$ and S_i be the event that $\mathsf{Exp}^{\mathsf{S\text{-}OTS}}_{\mathsf{SIG},\mathcal{F}}(n)$ outputs 1 in game $\mathbf{G}_i$:

Game $\mathbf{G}_0$: This is the original attack game for the strong one-time signature. It is trivial that $\Pr[\mathsf{S}_0] = \Pr[\mathsf{Exp}^{\mathsf{S\text{-}OTS}}_{\mathsf{SIG},\mathcal{F}}(n) = 1]$.

Game $\mathbf{G}_1$: When the adversary $\mathcal{F}$ outputs the forgery $(m^*, s^* = (r_1^*, c_2^*, r_2^*))$, we abort if $(m^*, r_2^*) \neq (m, r_2)$ but $c_2^* = c_2$, where m is $\mathcal{F}$'s one-time signing query and $s = (r_1, c_2, r_2)$ is the respond. It is clear to see the difference between $\mathbf{G}_1$ and $\mathbf{G}_0$ is bounded by the computational binding of the commitment scheme.

$$|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_0]| \leq \varepsilon.$$

Game $\mathbf{G}_2$: We simulate the signing query by using the trapdoor opening $\mathsf{Topen}(\mathsf{td}_2, (\cdot,\cdot), \cdot)$ on the second commitment and also change the key generation as follows:

- In the key generation, we compute $(\mathsf{vk}, \mathsf{sk})$ as in the real scheme. Additionally, we pick a random message $m' \leftarrow_{\$} \mathcal{M}$ and compute $(c_2, r_2') \leftarrow_{\$} \mathsf{Commit}(\mathsf{ck}_2, m')$. Keep (m', c_2, r_2') as secret and define $\mathsf{sk}' := (\mathsf{sk}, (m', c_2, r_2'))$.
- In the signing, upon receiving m, generate $r_2 \leftarrow_{\$} \mathsf{Topen}(\mathsf{td}_2, (m', r_2'), m)$ and compute r_1 as in the real scheme, $r_1 \leftarrow_{\$} \mathsf{Topen}(\mathsf{td}_1, (0, \hat{r}), c_2)$. Define the signature of m as $s := (r_1, c_2, r_2)$.

By the perfectly trapdoor opening, c_2 is the correct commitment of m, and formally $\mathsf{Open}(\mathsf{ck}_2, m, c_2, r_2) = 1$. Then $\mathbf{G}_2$ and $\mathbf{G}_1$ are identical.

$$\Pr[\mathsf{S}_2] = \Pr[\mathsf{S}_1]$$

Game $\mathbf{G}_3$: We continue to modify the key generation and signing query as follows:

- In the key generation, we compute $\hat{c}$ as $(\hat{c}, \hat{r}) \leftarrow_{\$} \mathsf{Commit}(\mathsf{ck}_1, c_2)$, instead of using the dummy message 0. And the rest are the same as in $\mathbf{G}_2$.
- In the signing, we generate r_2 as in $\mathbf{G}_2$ and define $r_1 := \hat{r}$.

The differences between $\mathbf{G}_3$ and $\mathbf{G}_2$ are: in $\mathbf{G}_3$ $\hat{c}$ is the commitment of c_2, while in $\mathbf{G}_2$ $\hat{c}$ is the commitment of 0. By the perfectly hiding, those are identical.

$$\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_2].$$

Moreover, $\mathbf{G}_3$ can be generated without using td_2. Assume ck_2 is given by the computational binding challenge. If the adversary $\mathcal{F}$ break the strong one-time signature, then we can break the computational binding of the commitment in the following way: once $\mathcal{F}$ outputs a forgery $(m^*, s^* = (r_1^*, c_2^*, r_2^*)) \neq (m, s = (r_1, c_2, r_2))$, if $(m^*, s^* = (r_1^*, c_2^*, r_2^*))$ is a valid forgery, then $((c_2^*, \hat{c}, r_1^*), (c_2, \hat{c}, r_1))$ is the correct answer for the computational binding challenge. The reasons are as follows:

- If $(m^*, r_2^*) \neq (m, r_2)$ then $c_2^* \neq c_2$ from $\mathbf{G}_1$. As a valid forgery, $\mathsf{Open}(\mathsf{ck}_1, c_2^*, \hat{c}, r_1^*) = \mathsf{Open}(\mathsf{ck}_1, c_2, \hat{c}, r_1) = 1$ holds. Then $((c_2^*, \hat{c}, r_1^*), (c_2, \hat{c}, r_1))$ is indeed the correct answer.
- If $(m^*, r_2^*) = (m, r_2)$ then $(c_2^*, r_1^*) \neq (c_2, r_1)$. It easy to see $((c_2^*, \hat{c}, r_1^*), (c_2, \hat{c}, r_1))$ is the correct answer.

Thus, we have $\Pr[\mathsf{S}_3] = \varepsilon$.

Combining all the above games, we have $\varepsilon' \leq 2\varepsilon$.

D.3 Security Proof of the UC PAKE (Theorem 4.1)

For the sake of simplicity, we give in Figure 11 an explicit version of the protocol described in Figure 5. We omit the additional verification that all the committed values are in the correct subsets, since in the proof below we will always easily guarantee this membership. As explained in Section 4, for the sake of simplicity, we denote by a unique ζ, and a unique z_i, the list of all the necessary challenges ζ, and their answers, for the bits of the password, in order to ensure that the probability that an adversary can gain something by cheating in $\mathcal{C}_i'$ is negligible (see details in the proof). Similarly, we mention a unique SPHF, omitting the details of its construction as a conjunction of SPHF.

The proof heavily relies on the properties of the commitments and smooth projective hash functions given in Section 4, 3 and Appendix B.

Initiator P_i		Receiver P_j
$(I1)$ $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{KeyGen}()$ $(\mathcal{C}_i, \mathcal{C}'_i) = k\text{-}\mathsf{DMPCom}'(\mathfrak{L}_i, \mathsf{ek}, W_i, 0; \mathbf{r}_i, \mathbf{r}'_i)$ $\chi = \mathfrak{H}_K{}'(\mathcal{C}'_i)$ $\mathcal{C}''_i = \mathsf{Trap.Commit}(\chi; \mathbf{t})$	$\xrightarrow{\mathsf{flow-one}}$ $(\mathrm{VK}_i, \mathcal{C}_i, \mathcal{C}''_i)$	
		$(R2)$ $\mathsf{Com}_j = \mathcal{C}_j = k\text{-}\mathsf{MP}(\mathfrak{L}_j, \mathsf{ek}, W_j; \mathbf{r}_j)$ $\zeta \xleftarrow{\$}$, $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L}'_i)$ $\mathsf{hp}_i = \mathsf{ProjKG}(\mathsf{hk}_i, \mathrm{L}'_i, \mathsf{Com}_i)$
	$\xleftarrow{\mathsf{flow-two}}$ $(\mathcal{C}_j, \zeta, \mathsf{hp}_i)$	
$(I3)$ $z_i = \mathbf{r}_i + \zeta \mathbf{r}'_i$, $\mathsf{Com}_i = \mathcal{C}_i + \zeta \mathcal{C}'_i$ $\mathsf{hk}_j \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L}'_j)$, $\mathsf{hp}_j = \mathsf{ProjKG}(\mathsf{hk}_j, \mathrm{L}'_j, \mathsf{Com}_j)$ $\sigma_i = \mathsf{Sign}(\mathrm{SK}_i, (\mathfrak{L}_i, \mathcal{C}_i, \mathcal{C}'_i, \mathcal{C}_j, \zeta, \mathsf{hp}_i, \mathsf{hp}_j))$ $H_i = \mathsf{Hash}(\mathsf{hk}_j, \mathrm{L}'_j, \mathfrak{L}_j, \mathsf{Com}_j)$ $H'_j = \mathsf{ProjHash}(\mathsf{hp}_i, \mathrm{L}_i, \mathfrak{L}_i, \mathsf{Com}_i; \mathbf{z})$ $\mathsf{sk}_i = H_i \oplus H'_j$ Sets the session as *accepted* and uses sk_i as a shared key	$\xrightarrow{\mathsf{flow-three}}$ $(\mathcal{C}'_i, \mathbf{t}, \mathsf{hp}_j, \sigma_i)$	
		$(R4)$ Abort if not $\mathsf{Verify}(\mathrm{VK}_i, (\mathfrak{L}_i, \mathcal{C}_i, \mathcal{C}'_i, \mathcal{C}_j, \zeta, \mathsf{hp}_i, \mathsf{hp}_j), \sigma_i)$ or not correct opening $\mathbf{t}$ for $\mathcal{C}'_i$ in $\mathcal{C}''_i$ $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\zeta}_i$ $H_j = \mathsf{Hash}(\mathsf{hk}_i, \mathrm{L}'_i, \mathfrak{L}_i, \mathsf{Com}_i)$ $H'_i = \mathsf{ProjHash}(\mathsf{hp}_j, \mathrm{L}_j, \mathfrak{L}_j, \mathsf{Com}_j; \mathbf{r}_j)$ $\mathsf{sk}_j = H'_i \oplus H_j$ Sets the session as *accepted* and uses sk_j as a shared key

Figure 11: Description of the password-authenticated key-exchange protocol for players (P_i, ssid), with index i, password W_i, random tape ω_i, label $\mathfrak{L}_i = (\mathfrak{L}, \mathrm{VK}_i)$, languages $\mathrm{L}_i = \mathrm{L}'_j = \{W_i\}$ and (P_j, ssid), with index j, password W_j, random tape ω_j, label $\mathfrak{L}_j = \mathfrak{L}$, languages $\mathrm{L}_j = \mathrm{L}'_i = \{W_j\}$. The label is $\mathfrak{L} = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$. The random values used in the commitments (witnesses) are all included in (r_i, r'_i) and r_j.

D.3.1 Sketch of Proof

The proof follows that of similar protocols [18, 2, 8]. In order to prove Theorem 4.1, we need to construct, for any real-world adversary $\mathcal{A}$ (controlling some dishonest parties), an ideal-world adversary $\mathcal{S}$ (interacting with dummy parties and the split functionality $s\mathcal{F}_{\mathrm{LAKE}}$) such that no environment $\mathcal{Z}$ can distinguish between an execution with $\mathcal{A}$ in the real world and $\mathcal{S}$ in the ideal world with non-negligible probability.

When initialized with security parameter k, the simulator first generates the CRS for the commitment (public parameters but also extraction and equivocation trapdoors). It then initializes the real-world adversary $\mathcal{A}$, giving it these values. The simulator then starts its interaction with the environment $\mathcal{Z}$, the functionality $\mathcal{F}_{\mathrm{pwKE}}$ and its subroutine $\mathcal{A}$.

Since we are in the static-corruption model, the adversary can only corrupt players before the execution of the protocol. We assume players to be honest or not at the beginning, and they cannot be corrupted afterwards. However, this does not prevent the adversary from modifying flows coming from the players. Indeed, since we are in a weak authenticated setting, when a player acts dishonestly (even without being aware of it), it is either corrupted, hence the adversary knows its private values and acts on its behalf; or the adversary tries to impersonate it with chosen/guessed inputs. In both cases, we say the player is $\mathcal{A}$-controlled. Following [18], we say that a flow is *oracle-generated* if it was sent by an honest player and arrives without any alteration to the player it was meant to. We say it is *non-oracle-generated* otherwise, that is if it was sent by a $\mathcal{A}$-controlled player (which means corrupted, or which flows have been modified by the adversary). The one-time signatures are aimed at avoiding changes of players during a session: if $\mathsf{flow-one}$ is oracle-generated for P_i, then $\mathsf{flow-three}$ cannot be non-oracle-generated without causing the protocol to fail because of the signature, for which the adversary does not know the signing key. On the other hand, if $\mathsf{flow-one}$ is non-oracle-generated for P_i, then $\mathsf{flow-three}$ cannot be oracle-generated without causing the protocol to fail, since the honest player would sign wrong flows (the flows the player sent before the adversary alters them). In both cases, the verifications of the signatures will fail at Steps $(R4)$ and P_j will abort. One can note that since there is one flow only in the protocol for P_j, its signature is not required.

D.3.2 Description of the Simulator

For the most part, the simulator simulates the protocol by executing it honestly on behalf of the honest parties, but using random dummy witnesses as inputs since the secret inputs (given to them by the environment) are unknown to the simulator. Furthemore, the commitment sent by P_i is equivocable, enabling the simulator to change its mind later on. And both commitments are extractable, enabling the simulator to recover the password used by the adversary. In the whole proof, in case the extraction fails, the simulator acts as if the simulation should fail. Indeed, the language of the smooth projective hash function not only verifies the equations, but also that the ciphertext is valid, and this verification will fail. More details follow.

We come back again to the case of our equivocable commitment with SPHF that is not a really extractable/binding commitment since the player can open it in a different way one would extract it, in case the second ciphertext does not encrypt to 0 values: if extraction leads to an inconsistent tuple, there is little chance that with the random ζ it becomes consistent; if extraction leads to a consistent tuple, there is little chance that with the random ζ it remains consistent, and then the real-life protocol will fail, whereas the ideal-one was successful at the TestPwd-time. But then, because of the positive TestPwd-answer, the NewKey-query takes the key-input into consideration, that is random on the initiator side because of the SPHF on an invalid word, and thus indistinguishable from the environment point of view from a failed session: this is a denial of service, the adversary should already be aware of.

We now describe the simulator in the three possible cases. During all these simulations, $\mathcal{S}$ knows the equivocability trapdoor of the trapdoor commitment and the decryption keys of the two encryption schemes.

Case 1: P_i is $\mathcal{A}$-controlled and P_j is honest. In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the honest player P_j.

STEP $(I1)$. This step is taken care of by the adversary, who sends its $\mathsf{flow-one}$, from which $\mathcal{S}$ extracts W_i.

STEP $(R2)$. The simulator asks a TestPwd query to the functionality to check whether P_j should have the password W_i (which means the protocol should succeed). In case of a success, $\mathcal{S}$ generates honestly the flow on behalf of P_j, in particular an encryption $\mathcal{C}_j$ on $W_j = W_i$. Otherwise, $\mathcal{S}$ produces a encryption $\mathcal{C}_j$ on a dummy W_j. It then generates a challenge value ζ and the hashing keys $(\mathsf{hk}_i, \mathsf{hp}_i)$ on $\mathcal{C}_i$. It sends the flow-two message $(\mathcal{C}_j, \zeta, \mathsf{hp}_i)$ to $\mathcal{A}$ on behalf of P_j.

STEP $(I3)$. This step is taken care of by the adversary, who sends its $\mathsf{flow-three}$.

STEP $(R4)$. Upon receiving $m = (\mathsf{flow-three}, \mathcal{C}'_i, \mathbf{t}, \mathsf{hp}_j, \sigma_i)$, $\mathcal{S}$ makes the verification checks, and possibly aborts. In case of correct checks, $\mathcal{S}$ already knows whether the protocol should succeed, thanks to the TestPwd query. If the protocol is a success, then $\mathcal{S}$ computes receiver session key honestly, and makes a NewKey to P_j. Otherwise, $\mathcal{S}$ makes a NewKey to P_j with a random key that will anyway not be used.

Case 2: P_i is honest and P_j is $\mathcal{A}$-controlled. In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the honest player P_i.

STEP $(I1)$. $\mathcal{S}$ generates a flow-one message by committing to a dummy password W_i and chooses a key pair $(\mathrm{SK}_i, \mathrm{VK}_i)$ for a one-time signature scheme. It gives this message $(\mathrm{VK}_i, \mathcal{C}_i, \mathcal{C}''_i)$ to $\mathcal{A}$ on behalf of (P_i, ssid).

STEP $(R2)$. This step is taken care of by the adversary, who sends its $\mathsf{flow-two} = (\mathcal{C}_j, \zeta, \mathsf{hp}_i)$, from which $\mathcal{S}$ extracts the committed password W_j.

STEP $(I3)$. $\mathcal{S}$ makes a TestPwd query to the functionality to know whether the password of P_i is indeed W_j (i.e. whether the protocol should succeed). In case of a success, $\mathcal{S}$ uses the equivocability trapdoor for each ζ to update the corresponding $\mathcal{C}'_i$ and $\mathbf{t}$ in order to contain the new consistent $W_i = W_j$ with respect to the challenge ζ. If the protocol should be a success, then $\mathcal{S}$ computes P_i's session key honestly, and makes a NewKey to P_i. Otherwise, $\mathcal{S}$ makes a NewKey to P_i with a random key that will anyway not be used. $\mathcal{S}$ sends the flow-three message $(\mathcal{C}'_i, \mathbf{t}, \mathsf{hp}_j, \sigma_i)$ to $\mathcal{A}$ on behalf of P_i, where σ_i is the signature on all the previous information.

STEP $(R4)$. This step is taken care of by the adversary.

Case 3: P_i and P_j are honest. In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the two honest players P_i and P_j. Since no player is controlled by $\mathcal{A}$, the TestPwd query will not provide any answer to the simulator. But thanks to the semantic security of the commitments, dummy values can be committed, no external adversary will make any difference.

STEP $(I1)$. $\mathcal{S}$ generates a flow-one message by committing to a dummy password W_i and chooses a key pair $(\mathrm{SK}_i, \mathrm{VK}_i)$ for a one-time signature scheme. It gives this message $(\mathrm{VK}_i, \mathcal{C}_i, \mathcal{C}''_i)$ to $\mathcal{A}$ on behalf of (P_i, ssid).

STEP $(R2)$. $\mathcal{S}$ generates an encryption $\mathcal{C}_j$ on a dummy password W_j. It then generates a challenge value ζ and the hashing keys $(\mathsf{hk}_i, \mathsf{hp}_i)$ on $\mathcal{C}_i$. It sends the flow-two message $(\mathcal{C}_j, \zeta, \mathsf{hp}_i)$ to $\mathcal{A}$ on behalf of P_j.

STEP $(I3)$. $\mathcal{S}$ makes a NewKey to P_i with a random key that will anyway not be used, since no player is corrupted. $\mathcal{S}$ sends the flow-three message $(\mathcal{C}'_i, \mathbf{t}, \mathsf{hp}_j, \sigma_i)$ to $\mathcal{A}$ on behalf of P_i, where σ_i is the signature on all the previous information.

STEP $(R4)$. Upon receiving $m = (\mathsf{flow-three}, \mathcal{C}'_i, \mathbf{t}, \mathsf{hp}_j, \sigma_i)$ from its peer session $(P_i; \mathrm{ssid})$, the signature is necessarily correct. $\mathcal{S}$ makes a NewKey to P_j with a random key that will anyway not be used, since no player is corrupted.

D.3.3 Description of the Games

Game $\boldsymbol{G}_0$: This is the real game, where every flow from honest players are generated correctly by the simulator which knows the inputs sent by the environment to the players. There is no use of the ideal functionality for the moment.

Game $\boldsymbol{G}_1$: In this game, the simulator knows the decryption key for $\mathcal{C}_i$ when generating the CRS. But this game is almost the same as the previous one except the way sk_j is generated when P_i is corrupted and P_j honest. In all the other cases, the simulator does as in $\boldsymbol{G}_0$ by playing honestly (still knowing its private values). When P_i is corrupted and P_j honest, $\mathcal{S}$ does as before until $(R4)$, but then, it extracts the values committed to by the adversary in Com_i (using the decryption key for $\mathcal{C}_i$) and checks whether the password is consistent with the value sent to P_j by the environment. If the passwords are not consistent (or decryption rejects), P_j is given a random session key sk_j. This game is statistically indistinguishable from the former one thanks to the smoothness of the SPHF on Com_i.

Game $\boldsymbol{G}_2$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS. This game is almost the same as the previous one except that $\mathcal{S}$ extracts the values committed to by the adversary in $\mathcal{C}_i$ to check consistency of the passwords, and does not wait until Com_i. If the passwords are not consistent (or decryption rejects), P_j is given a random session key sk_j.

The game is indistinguishable from the previous one except if Com_i contains consistent values whereas $\mathcal{C}_i$ does not, but because of the unpredictability of the $\mathfrak{n}$ challenges ζ, and the trapdoor commitment that is computationally binding under the SIS problem, the probability is bounded by $1/2^{\mathfrak{n}}$.

The distance between the two games is thus bounded by the probability to break the binding property of the trapdoor commitment.

Game $\boldsymbol{G}_3$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS, as in $\boldsymbol{G}_2$. Actually, in the above game, when P_i is corrupted and P_j honest, if the extracted password from $\mathcal{C}_i$ is not consistent, P_j does not have to compute hash values. The random coins are not needed anymore. In this game, in this particular case, $\mathcal{S}$ generates $\mathcal{C}_j$ with a dummy password $\widetilde{W_j}$.

This game is computationally indistinguishable from the former one thanks to the $\mathsf{IND\text{-}CPA}$ property of the encryption scheme involved in $\mathcal{C}_j$. To prove this indistinguishability, one makes q hybrid games, where q is the number of such sessions where P_i is corrupted and P_j is honest but extracted languages from $\mathcal{C}_i$ are not consistent with inputs to P_j. More precisely, in the k-th hybrid game G_k (for $1 \leq k \leq q$), in all such sessions before the k-th one, $\mathcal{C}_j$ is generated by encrypting $\widetilde{W_j}$, in all sessions after the k-th one, $\mathcal{C}_j$ is generated by encrypting W_j, and in the k-th session, $\mathcal{C}_j$ is generated by calling the left-or-right encryption oracle on $(W_j, \widetilde{W_j})$. It is clear that the game $\boldsymbol{G}_2$ correponds to G_1 with the "left" oracle, and the game $\boldsymbol{G}_3$ corresponds to G_q with the "right" oracle. And each time, G_k with the right oracle is identical to G_{k+1} with the "left" oracle, while every game G_k is an $\mathsf{IND\text{-}CPA}$ game. It is possible to use the encryption oracle because the random coins are not needed in these sessions.

Game $\boldsymbol{G}_4$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS, as in $\boldsymbol{G}_2$. Now, when P_i is corrupted and P_j honest, if the extracted password from $\mathcal{C}_i$ is consistent, $\mathcal{S}$ knows W_j (the same as the value W_i sent by the environment). It uses it to generate the ciphertext $\mathcal{C}_j$. $\mathcal{S}$ can compute the correct value sk_j from the random coins, and gives it to P_j.

This game is perfectly indistinguishable from the former one.

Note that the value sk_j computed by $\mathcal{S}$ can be computed by the adversary if the latter indeed sent a valid password W_i in $\mathcal{C}_i$ (that is not explicitly checked in this game). Otherwise, sk_j looks random from the smoothness of the SPHF. As a consequence, in this game, sessions where P_i is corrupted and P_j is honest look ideal, while one does not need anymore the inputs from the environment sent to P_j to simulate honest players.

Game $\boldsymbol{G}_5$: We now consider the case where P_i is honest. The simulator has to simulate P_i behavior. To do so, it will know the equivocability trapdoor for the trapdoor commitment. But for other cases, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS. In $(I1)$, the simulator still encrypts W_i from the environment to produce $\mathcal{C}_i$. It chooses at random a dummy value $\mathcal{C}'_i$ and computes honestly the equivocable commitment $\mathcal{C}''_i$, knowing the random value $\mathbf{t}_i$. In $(I3)$, after receiving ζ from P_j, it chooses random coins $\mathbf{z}_i$ and computes Com_i as the encryption of W_i with the random coins $\mathbf{z}_i$. (Since this is a double encryption scheme, it uses the redundancy from $\mathcal{C}_i$: namely for k-DMPCom, it uses $\mathbf{T}$ from $\mathcal{C}_i$). Thanks to the homomorphic property, it can compute $\mathcal{C}'_i$ as $(\mathsf{Com}_i - \mathcal{C}_i)/\zeta$, and equivocate $\mathcal{C}"_i$. $\mathcal{C}'_i$ should be an encryption of 0 under the random coins $\mathbf{r}'_i$ that are implicitly defined, but unknown. Thanks to the properties of the different commitments recalled in Appendix B, and the perfect-hiding property of the trapdoor commitment, this is a perfect simulation. It then computes the hash values honestly, using $\mathbf{z}_i$.

Game $\boldsymbol{G}_6$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the trapdoor commitment when generating the CRS. When P_i is honest, $\mathcal{S}$ generates the commitment $\mathcal{C}_i$ by choosing a dummy password $\widetilde{W_i}$ instead of W_i. Everything else is unchanged from $\boldsymbol{G}_5$.

This game is thus indistinguishable from the former one thanks to the IND-CCA property of the encryption scheme involved in $\mathcal{C}_i$. As for the proof of indistinguishability of Game $\boldsymbol{G}_3$, we do a sequence of hybrid games, where $\mathcal{C}_i$ is generated be either encrypting W_i or $\widetilde{W_i}$, or asking the left-or-right oracle on $(W_i, \widetilde{W_i})$. We replace the decryption key for $\mathcal{C}_i$ by access to the decryption oracle on $\mathcal{C}_i$. Then, one has to take care that no decryption query is asked on one of the challenge ciphertexts involved in the sequence of games. This would mean that the adversary would replay in another session a ciphertext oracle-generated in another session. Because of the label which contains the verification key oracle-generated, one can safely reject the ciphertext.

Game $\boldsymbol{G}_7$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the trapdoor commitment when generating the CRS. When P_i is honest, $\mathcal{S}$ generates the commitment $\mathcal{C}_i$ by choosing a dummy password $\widetilde{W_i}$. It then computes $\mathcal{C}'_i$ by encrypting the value $(W_i - \widetilde{W_i})/\zeta$ with randomness $\mathbf{z}_i - \mathbf{r}_i/\zeta$. This leads to the same computations of $\mathcal{C}_i$ and $\mathcal{C}'_i$ as in the former game. The rest is done as above.

This game is perfectly indistinguishable from the former one.

Game $\boldsymbol{G}_8$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the trapdoor commitment when generating the CRS. When P_i and P_j are both honest, if the words and languages are correct, players are both given the same random session key $\mathsf{sk}_i = \mathsf{sk}_j$. If the passwords are not compatible, random independent session keys are given.

Since the initiation flow $I1$ contained an oracle-generated verification key, unless the adversary managed to forge signatures, all the flows are oracle-generated. First, because of the pseudo-randomness of the SPHF, H_i is unpredictable, and independent of H'_j, hence sk_i looks random. Then, if the passwords are compatible, we already have $\mathsf{sk}_j = \mathsf{sk}_i$ in the previous game. However, if they are not compatible, either H'_i is independent of H_i, or H'_j is independent of H_j, and in any case, sk_j where already independent of sk_i in the previous game. This game is thus computationally indistinguishable from the former one, under the pseudo-randomness of the two SPHF.

Game $\boldsymbol{G}_9$: In this above game, the hash values do not have to be computed anymore when P_i and P_j are both honest. The random coins are not needed anymore.

In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the trapdoor commitment when generating the CRS. When P_i and P_j are both honest, $\mathcal{S}$ generates $\mathcal{C}'_i$ and $\mathcal{C}_j$ with dummy values $\widetilde{W_i}$ and $\widetilde{W_j}$. In this game, sessions where P_i and P_j are both honest look ideal, while one does not need anymore the inputs from the environment sent to P_i and P_j to simulate honest players.

This game is computationally indistinguishable from the former one thanks to the IND-PD-CCA and IND-CPA properties of the encryption schemes involved in $\mathcal{C}'_i$ and $\mathcal{C}_j$. For the proof on indistinguishability between the two games, we make two successive sequences of hybrid games, as for the proof of indistinguishability of Game $\boldsymbol{G}_3$. One with the IND-PD-CCA game: a sequence of hybrid games, where $\mathcal{C}_i$ is generated by encrypting $\widetilde{W_i}$, and $\mathcal{C}'_i$ by encrypting either W_i or $\widetilde{W_i}$, but in the critical session, one asks for the left-or-right oracle Encrypt on $(\widetilde{W_i}, \widetilde{W_i})$, and the left-or-right oracle Encrypt$'$ on $(\widetilde{W_i}, \widetilde{W_i})$. The decryption key for $\mathcal{C}_i$ is replaced by an access to the decryption oracle on $\mathcal{C}_i$. As above, one has to take care that no decryption query is asked on a challenge ciphertext $\mathcal{C}'_i$, but the latter cannot be valid since it is computed from $\mathcal{C}_i$ values not controlled by the adversary. The second hybrid sequence uses IND-CPA games on $\mathcal{C}_j$ exactly as in the proof of indistinguishability of Game $\boldsymbol{G}_3$.

Game $\boldsymbol{G}_{10}$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the trapdoor commitment when generating the CRS, but also the decryption key for $\mathcal{C}_j$. When P_i is honest and P_j corrupted, $\mathcal{S}$ extracts the password committed to by the adversary in $\mathcal{C}_j$. It checks whether it is consistent with the password sent to P_i by the environment. If the passwords are not consistent (or decryption rejects), P_i is given a random session key sk_i.

This game is statistically indistinguishable from the former one thanks to the smoothness of the SPHF.

Game $\boldsymbol{G}_{11}$: In this game, the simulator still knows the decryption keys for $\mathcal{C}_i$ and $\mathcal{C}_j$ and the equivocability trapdoor for the trapdoor commitment when generating the CRS.

In the above game, when P_i is honest and P_j corrupted, if the extracted password from $\mathcal{C}_j$ is not consistent, P_i does not have to compute hash values. The random coins are not needed anymore. In this game, in this particular case, $\mathcal{S}$ generates $\mathcal{C}'_i$ with a dummy random $\widetilde{W_i}$.

This game is computationally indistinguishable from the former one thanks to the IND-PD-CCA property of the encryption scheme involved in $\mathcal{C}'_i$. The proof uses the same sequence of hybrid games with the IND-PD-CCA game on $(\mathcal{C}_i, \mathcal{C}'_i)$ as in the proof of indistinguishability of Game $\boldsymbol{G}_9$.

Game $\boldsymbol{G}_{12}$: In this game, the simulator still knows the decryption keys for $\mathcal{C}_i$ and $\mathcal{C}_j$ and the equivocability trapdoor for the trapdoor commitment when generating the CRS. Now, when P_i is honest and P_j corrupted, if the extracted password from $\mathcal{C}_j$ is consistent, $\mathcal{S}$ knows W_i (the same as the value W_j sent by the environment). It uses it to generate the ciphertext $\mathcal{C}'_i$. $\mathcal{S}$ can compute the correct value sk_i from the random coins, and gives it to P_i. In this game, sessions where P_i is honest and P_j is corrupted look ideal, while one does not need anymore the inputs from the environment sent to P_i to simulate honest players.

This game is perfectly indistinguishable from the former one.

Game $\boldsymbol{G}_{13}$: In this game, $\mathcal{S}$ now uses the ideal functionality: NewSession-queries for honest players are automatically forwarded to the ideal functionality, for corrupted players, they are done by $\mathcal{S}$ using the values extracted from $\mathcal{C}_i$ or $\mathcal{C}_j$. In

order to check consistency of the passwords, $\mathcal{S}$ asks for a $\mathsf{TestPwd}$. When one player is corrupted, it learns the outcome: success or failure. It can continue the simulation in an appropriate way.

D.4 Security Proof of the UC Commitment (Theorem 5.1)

We now provide a full proof, with a sequence of games, that the protocol presented on Figure 7 emulates the ideal functionality $\mathcal{F}_{\mathsf{mcom}}$ against adaptive corruptions with erasures. This sequence starts from the real game, where the adversary interacts with real players, and ends with the ideal game, where we have built a simulator that makes the interface between the ideal functionality and the adversary.

We denote by $C_3 = C_1 + \zeta C_2$, the tuple involved in the last check. It should be a partial encryption of x under randomness $\mathbf{z} = \mathbf{r} + \zeta \mathbf{s}$ and noise $\mathbf{e}_z = \mathbf{e}_r + \zeta \mathbf{e}_s : C_3 = \mathsf{MP}^*(x; \mathbf{T}, \mathbf{z}, \mathbf{e}_z)$.

Game G_0 : This is the real game, in which every flow from the honest players is generated correctly by the simulator which knows the input x sent by the environment to the sender. There is no use of the ideal functionality for the moment.

Game G_1 : In this game, we focus on the simulation of an honest receiver interacting with a corrupted sender. Executions with an honest sender are still simulated as before, using the input x. The simulator will generate the CRS in such a way it knows the Micciancio Peikert decryption key, but not the trapdoor for the commitment.

Upon receiving the values (c_t^1, c_t^2) from the adversary, the simulator simply chooses a challenge ζ at random and sends it to the adversary, as P_j would do with P_i. After receiving the values $(C_1, \mathbf{t}_1)$, the simulator checks the consistency of the Trapdoor commitment c_t^1 and aborts in case of failure. It then uses the decryption key to recover the value x' sent by the adversary. In case of invalid ciphertext, one sets $x' = \perp$. It stores $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, C_1, \zeta, c_t^2)$ and $(x', \mathsf{sid}, P_i, P_j)$ (this will correspond later to the Commit query to the ideal functionality, in the ideal game). Upon receiving the values $(x, C_2, \mathbf{t}_2, \mathbf{z}, e_z)$, the simulator does as P_j would do in checking the commitment c_t^2 and that $C_3 = \mathsf{MP}^*(x; \mathbf{T}, \mathbf{z}, e_z)$, but accepts x' as the opening for the commitment.

The only difference with the previous game is that P_i will accept x', as decrypted from $C_1 = \mathsf{MP}(x'; \mathbf{T}, \mathbf{r})$, for the decommitment instead of the value x output at the decommitment time, which matches with $C_3 = \mathsf{MP}^*(m; \mathbf{T}, \mathbf{z}, e_z)$, but that is also contained in c_t^2 together with C_2. We will show that under the binding property of the Trapdoor commitment, one has $x' = x$ with overwhelming probability, and thus there is no real difference.

Let us assume that $x' \neq x$ in at least one of such executions: for the first one, we rewind the adversary to the step 4., and send a new challenge ζ_2. Then the adversary should send the same C_1, otherwise one directly breaks the binding property of the Trapdoor commitment or finds a collision for $\mathfrak{H}_K$, and the same pair (x, C_2) in the decommit phase for the same reason, but possibly different $\mathbf{z}_2$.

If ζC_2 decrypts to 0: Writing $C_3' = \mathbf{z}^\top (A + (0|\mathbf{T}G)) + xq/2 + \mathbf{e}_r + (\alpha + \zeta\beta) + \zeta \mathbf{e}_s$, for α, β such that $|\zeta\beta| < q/4$ and $|\alpha| < q/4$ we have $|\alpha + \zeta\beta| > q/4$, this means that with $\zeta_2 = -\zeta$ the new C_3' decrypts to x so $x = x'$ as $|\alpha + \zeta_2\beta| < q/4$.

The same study can be done if ζC_2 decrypts to 1.

We stress that the rewind here is just for the proof of indistinguishability of the two games, but not in the simulation.

In case of corruption of the receiver, one can note that he has no secret.

Game G_2 : In this game, we start modifying the simulation of an honest sender, still knowing his input x. For the honest verifier against a corrupted sender, we still have to know the Micciancio Peikert decryption key to run the same simulation as in the previous game. But we now need to know the $\aleph$ trapdoor used for equivocating the trapdoor commitment.

This game is almost the same as the previous one excepted the way the double ciphertext is generated: $(C_1, C_2) = \mathsf{DMP}(x, y; \mathbf{r}, \mathbf{s})$, for a random y instead of 0. The rest of the commit phase is unchanged.

At the decommit phase, $\mathcal{S}$ chooses random coins $\mathbf{z}, \mathbf{e}_z$ and computes $C_3 = \mathsf{MP}^*(x; \mathbf{T}, \mathbf{z}, \mathbf{e}_z)$, and then "repairs" $C_2 = (C_3 - C_1)/\zeta$, and $\mathbf{t}_2$ for being able to open c_t^2 to this new value.

Thanks to the homomorphic property, the repaired C_2 is similar to a correct ciphertext of 0, and the equivocation of the Trapdoor commitment guarantees a correct opening. This game is thus perfectly indistinguishable from the previous one.

In case a corruption of P_i occurs before the decommit phase, the simulator anticipates the equivocation of c_t^2.

Game G_3 : One can note that in the previous game, $\mathbf{r}, \mathbf{e}_r$ is not used anymore to compute $\mathbf{z}, \mathbf{e}_z$. One could thus ignore it, unless $\mathcal{P}_i$ gets corrupted before ζ has been sent, since we should be able to give it. But in such a case, one can compute again C_1 knowing $\mathbf{r}, \mathbf{e}_r$ and equivocate c_t^1. We then alter again the way the double ciphertext is generated: $(C_1, C_2) = \mathsf{DMP}(x', y; \mathbf{r}, \mathbf{s})$, for random x' and y. Everything remains unchanged.

The unique change is thus the ciphertext C_1 that encrypts a random x' instead of x. One can run the $\mathsf{IND\text{-}CCA}$ security game, in an hybrid way, to show this game is indistinguishable from the previous one. To this aim, one has to show that the random coins $\mathbf{r}$ are not needed to be known, and that the challenge ciphertexts are never asked for decryption (where the decryption key here is replaced by an access to the decryption oracle, hence the $\mathsf{IND\text{-}CCA}$ security game). The former point has been discussed above. For the latter, we have shown that the value actually encrypted in C_1 by the corrupted sender is the value sent at the decommit phase, which would even break the one-wayness of the encryption. Hence, if such a replay happens, one knows that the decommit phase will fail.

In case of corruption of P_i before receiving ζ, Trapdoor commitments only have been sent, and they can thus be equivocated with correct values (given by either the ideal functionality or the adversary). In case of corruption of P_i after having received ζ, one does has before, anticipating the equivocation of c_t^2.

Game G_4 : This is the ideal game, in which the simulator works as described below: when P_i is corrupted, one uses the decryption of C_1 to send the Commit query to the ideal functionality, when P_i is honest one can wait for the receipt and reveal confirmations from the adversary to conclude the simulation of the real flows.

D.4.1 Description of the Simulator

Setup. The simulator generates the parameters, knowing the Micciancio Peikert decryption key and the equivocation trapdoor.

When P_i is honest.

COMMIT STAGE: Upon receiving the information that a commitment has been performed, with $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from $\mathcal{F}_{\mathsf{mcom}}$, $\mathcal{S}$ computes $(C_1, C_2) = \mathsf{DMP}(x', y; \mathbf{r}, \mathbf{s})$, for random x' and y but then follows as P_i would do. If P_j is honest too, one just has to send a random ζ.

In case of corruption of P_i before receiving ζ, one can equivocate c_t^1, otherwise one equivocates c_t^2, as explained above, in both cases using the value given either by the ideal functionality or the adversary, according to the time of the corruption.

DECOMMIT STAGE: Upon receiving the information that the decommitment has been performed on x, with $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$ from $\mathcal{F}_{\mathsf{mcom}}$, $\mathcal{S}$ exploits the equivocability of the Trapdoor commitment: it first chooses a random $\mathbf{z}, e_z$ and computes the ciphertext $C_3 = \mathsf{MP}^*(x; \mathbf{T}, \mathbf{z}, e_z)$. It then adapts $C_2 = (C_3 - C_1)/\zeta$ and uses the trapdoor for the commitment to produce a new value $\mathbf{t}_2$ corresponding to the new value C_2. It then simulates the decommit phase to P_j.

When P_i is corrupted and P_j is honest.

COMMIT STAGE: Upon receiving $(C_1, \mathbf{t}_1)$ from the adversary, $\mathcal{S}$ decrypts the ciphertext C_1 and extracts x. If the decryption is invalid, $\mathcal{S}$ sends $(\mathsf{Commit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \perp)$ to $\mathcal{F}_{\mathsf{mcom}}$. Otherwise, $\mathcal{S}$ sends $(\mathsf{Commit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$.

DECOMMIT STAGE: $\mathcal{S}$ acts as a regular honest user P_j from the incoming message of $\mathcal{A}$ on behalf of P_i. In case of validity, it sends the query $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid})$.

E Cheat Sheet, Pairing-based and Lattice-Based Cryptography

In Figure 12, we compare pairing-based encryptions, and their equivalent over Lattices. We recall the parameters for each of those construction

- Dual Regev SPHF parameters (section 3.1): $n = \mathrm{poly}(\lambda)$, $q = 2^{\Theta(n)}$, $m \geq \Theta(nk)$, and $s = q/f(n)$ where f is superpolynomial but subexponential, in particular $s = q \cdot \mathsf{negl}(n)$.
- Micciancio Peikert SPHF parameters (section 3.2): $n = \mathrm{poly}(\lambda)$ and it is a power of 2, $q = 2^{\Theta(n)}$, $k = \lceil \log q \rceil = O(\log n)$, $\bar{m} = \Theta(nk)$, $m = \bar{m} + nk$ and finally $s = q/f(n)$ where f is superpolynomial but subexponential, in particular $s = q \cdot \mathsf{negl}(n)$.

	ElGamal [24]	Dual Regev [30]
Setup	$(p, \mathbb{G}, g)$	$(n, m, q, s, \bar{\mathbf{A}} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n\times m}))$
$\mathsf{KG}(1^n)$	$\mathsf{dk} = (t \leftarrow \mathcal{U}(\mathbb{Z}_p)\vert -1)^\top$	$\mathsf{dk} = (\mathbf{t} \leftarrow \mathcal{U}(\{0,1\}^m)\vert -1)$;
	$\mathsf{ek} = A = (g\vert g^t)$	$\mathsf{ek} = \mathbf{A} = [\,\bar{\mathbf{A}}\vert(\bar{\mathbf{A}}\cdot\mathbf{t})\,] \in \mathbb{Z}_q^{n\times m+1}$
$\mathsf{Encrypt}(\mathsf{ek}, M; s)$	$M \in \mathbb{G}, s \leftarrow \mathcal{U}(\mathbb{Z}_p)$	$M \in \{0,1\}, \mathbf{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n), \mathbf{e} \leftarrow D_{\mathbb{Z},s}^{m+1}$
	$\mathbf{c} = (A^\top)^s + (0\vert M)^\top$	$\mathbf{c} = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e} + (\mathbf{0}\vert\mathsf{ECC}(M))^\top \in \mathbb{Z}_q^{m+1}$
$\mathsf{Decrypt}(\mathsf{dk}, \mathbf{c})$	$M' = \mathbf{c}^{\mathsf{dk}}$	$M' = \mathsf{ECC}^{-1}(\langle \mathsf{dk}, \mathbf{c}\rangle) \in \{0,1\}$
CPA-Security	Under DDH	Under LWE
SPHF:		
HashKG	$\mathsf{hk} \leftarrow \mathcal{U}(\mathbb{Z}_p^2)$	$\mathsf{hk} \leftarrow D_{\mathbb{Z},\omega(\sqrt{\log\lambda})}^{m+1}$
$\mathsf{ProjKG}(\mathsf{ek}, \mathsf{hk})$	$\mathsf{hp} = A^{\mathsf{hk}}$	$\mathsf{hp} = \mathbf{A} \cdot \mathsf{hk}$
$\mathsf{ProjHash}(\mathsf{hp}, s)$	$H' = \mathsf{hp}^s$	$H' = \mathsf{ECC}^{-1}(\langle \mathsf{hp}, \mathbf{s}\rangle)$
$\mathsf{Hash}(\mathbf{c}, M', \mathsf{hk})$	$H = (\mathbf{c} - (0\vert M)^\top)^{\mathsf{hk}}$	$H = \mathsf{ECC}^{-1}\left(\langle \mathsf{hk}, \mathbf{c} - (\mathbf{0}\vert\mathsf{ECC}(M')^\top\rangle\right)$

	Cramer Shoup[21]	Micciancio Peikert
Setup	$(p, \mathbb{G}, \bar{\mathbf{A}} = (g_1, g_2), \mathcal{H})$	$(n, m, q, s, \bar{\mathbf{A}})$
$\mathsf{KG}(1^n)$	$\mathsf{dk}_1 = (1, -z \xleftarrow{\$} \mathbb{Z}_p, 0, 0)^\top$	
	$\mathsf{dk}_2 = \mathbf{R}_1, \mathbf{R}_2 = (x_1, x_2)^\top, (y_1, y_2)^\top \leftarrow \mathbb{Z}_p^2$	$\mathsf{dk} = (\mathbf{R}\vert\mathbf{1})$;
	$\mathsf{ek} = A, G = (g_1^t\vert\bar{\mathbf{A}}\vert\bar{\mathbf{A}}^{\mathbf{R}_1}), \bar{\mathbf{A}}^{\mathbf{R}_2}$	$\mathsf{ek} = (\mathbf{A} = \left[\bar{\mathbf{A}}\vert - \mathbf{R}\bar{\mathbf{A}}\right]), \mathbf{G})$
$\mathsf{Encrypt}(\mathsf{ek}, M; s)$	$M \in \mathbb{G}, s \in_R \mathbb{Z}_p$	$M \in \{0,1\}, s \leftarrow \mathcal{U}(\mathbb{Z}_q^n), e \leftarrow D_{\mathbb{Z},s}^m, \mathbf{T} \leftarrow \mathcal{T}$
	$\mathbf{c} = (()A + (\mathbf{0}\vert G^\xi))^\top)^s + (M\vert\mathbf{0})^\top$	$\mathbf{c} = (\mathbf{A} + (\mathbf{0}\vert\mathbf{TG}))^\top\mathbf{s} + \mathbf{e} + (\mathbf{0}\vert\mathsf{ECC}(M))^\top$
$\mathsf{Decrypt}(\mathsf{dk}, \mathbf{c})$	$C^{(0\vert\mathsf{dk}_{2,1}\vert\xi\mathsf{dk}_{2,2}\vert-1)} \stackrel{?}{=} 1_{\mathbb{G}}$	$\mathsf{Verify}(\mathsf{OTS})$
	$M' = \mathbf{c}^{\mathsf{dk}_1}$	$M' = \mathsf{ECC}^{-1}(\langle \mathsf{dk}_T, \mathbf{c}\rangle)$
CCA-Security	Under DDH	Under LWE
SPHF:		
HashKG	$\mathsf{hk} = u, \mathbf{v} \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p^3$	$\mathsf{hk} = \mathbf{v} \leftarrow D_{\mathbb{Z},\omega(\sqrt{\log\lambda})}^m$
$\mathsf{ProjKG}(\mathsf{ek}, \mathsf{hk})$	$\mathsf{hp} = (A^{\mathbf{v}}, (A_1\vert\mathbf{0}\vert G)^{(u\vert\mathbf{0}\vert\mathbf{v}_4)})$	$\mathsf{hp} = (\mathbf{A} \cdot \mathbf{v}, (\bar{\mathbf{0}}\vert\mathbf{G}) \cdot (\mathbf{0}\vert\mathbf{v}_*))$
$\mathsf{ProjHash}(\mathsf{hp}, s)$	$H' = \mathsf{hp}_1^s.\mathsf{hp}_2^{\xi s}$	$H' = \mathsf{ECC}^{-1}(\langle \mathsf{hp}_1, \mathbf{s}\rangle + \langle \mathsf{hp}_2, \mathbf{T}^\top s\rangle)$
$\mathsf{Hash}(\mathbf{c}, M', \mathsf{hk})$	$\mathcal{C} = \mathbf{c} - (M'\vert 0)^\top$	$\mathcal{C} = \mathbf{c} - (\mathbf{0}\vert\mathsf{ECC}(M'))^\top$
	$H = \mathcal{C}^{\mathbf{v}+\xi(u\vert\mathbf{0})}$	$H = \mathsf{ECC}^{-1}\left(\langle \mathsf{hk}, \mathcal{C}\rangle\right)$

Figure 12: Parallel comparison between classical encryptions on pairing and lattices, and the associated SPHF