Hash Proof Systems over Lattices Revisited

Fabrice Benhamouda[1], Olivier Blazy[2♯], Léo Ducas[3♭], and Willy Quach[4♪]

[1] IBM Research, Yorktown Heights, USA
fabrice.benhamouda@normalesup.org
[2] XLim, Université de Limoges, France
olivier.blazy@unilim.fr
[3] CWI, Amsterdam, The Netherlands
leo.ducas@cwi.nl
[4] Northeastern University, USA
willy.quach@ens-lyon.fr

Abstract. Hash Proof Systems or Smooth Projective Hash Functions (SPHFs) are a form of implicit arguments introduced by Cramer and Shoup at Eurocrypt'02. They have found many applications since then, in particular for authenticated key exchange or honest-verifier zero-knowledge proofs. While they are relatively well understood in group settings, they seem painful to construct directly in the lattice setting.

Only one construction of an SPHF over lattices has been proposed in the standard model, by Katz and Vaikuntanathan at Asiacrypt'09. But this construction has an important drawback: it only works for an ad-hoc language of ciphertexts. Concretely, the corresponding decryption procedure needs to be tweaked, now requiring q many trapdoor inversion attempts, where q is the modulus of the underlying Learning With Errors (LWE) problem.

Using harmonic analysis, we explain the source of this limitation, and propose a way around it. We show how to construct SPHFs for standard languages of LWE ciphertexts, and explicit our construction over a tag-IND-CCA2 encryption scheme à la Micciancio-Peikert (Eurocrypt'12). We then improve our construction and our analysis in the case where the tag is known in advance or fixed (in the latter case, the scheme is only IND-CPA) with a super-polynomial modulus, to get a stronger type of SPHF, which was never achieved before for any language over lattices.

Finally, we conclude with applications of these SPHFs: password-based authenticated key exchange, honest-verifier zero-knowledge proofs, and a relaxed version of witness encryption.

Keywords. Hash Proof Systems, SPHF, Lattices, Learning With Errors, Harmonic Analysis.

[♯] This work has been supported by the French ANR project ID-FIX (ANR-16-CE39-0004).

[♭] This work has been supported by a Veni Grant from NWO.

[♪] Research supported in part by NSF grants CNS-1314722, CNS-1413964. This work was partly realized during an internship program at CWI.

1 Introduction

Harmonic analysis is a powerful tool in geometry of numbers, especially in combination with Gaussian measure, which has lead to important progress on transference theory [Ban93]. Those tools also played a crucial role for the foundation of lattice-based cryptography, being at the heart of proofs of worst-case hardness for lattice problems, such as the Short Integer Solution problem (SIS) and the Learning with Errors (LWE) problem [MR04, Reg05, GPV08]. Later, security proofs relied on a few convenient lemmas in a black-box manner, and for most applications this was sufficient: lattice-based cryptography quickly caught up with pairing-based cryptography, for example with the constructions of (Hierarchical) Identity Based Encryption's [GPV08, CHKP10, MP12] and beyond [Boy13, GVW13, GVW15].

There nevertheless remains one primitive for which lattice-based cryptography is still far behind: Hash Proof Sytems or Smooth Projective Hash Functions (SPHFs) [CS02]. Beyond the original Chosen-Ciphertext secure encryption scheme of Cramer and Shoup [CS98], SPHFs give rise to generalized classes of Authenticated Key Exchange (Password-based, Language-based, . . .) [GL06, ACP09, KV11, BBC$^+$13a]. They also have been used in Oblivious Transfer [Kal05, HK12], One-Time Relatively-Sound Non-Interactive Zero-Knowledge Arguments [JR12], and Zero-Knowledge Arguments [BBC$^+$13b].

An SPHF can be seen as an implicit (designated-verifier) zero-knowledge proof for a language. The most useful languages for SPHFs are the languages of ciphertexts of a given plaintext M.

To our knowledge, there is only one construction of SPHF for a lattice-based encryption scheme in the standard model, given by Katz and Vaikuntanathan [KV09]. There is also a subsequent work by Zhang and Yu who propose an interesting new lattice-based SPHF in [ZY17]. But the language of the SPHF relies on simulation-sound non-interactive zero-knowledge proofs which we do not know how to construct just under lattice-based assumptions without random oracle.

Unfortunately, the only standard-model lattice-based SPHF construction in [KV09] has a main drawback: the language of the SPHF is not simply defined as the set of valid standard LWE ciphertexts. Naturally, the set of valid ciphertexts of 0 should correspond to the set of ciphertexts close to the lattice defined by the public key. Instead, their language includes all the ciphertexts $\boldsymbol{c}$ such that at least one integer multiple is close to the public lattice. This makes the decryption procedure very costly (about q trapdoor inversions), and forbids the use of super-polynomial modulus q. This limitation is a serious obstacle to the construction of a stronger type of SPHF introduced in [KV11], namely *word-independent* SPHF for which the *projection key* (which can be seen as the public key of the SPHF) does not depend on the ciphertext $\boldsymbol{c}$ (a.k.a., word in the SPHF terminology).[5]

[5] Word-independent SPHFs are also called KV-SPHF in [BBC$^+$13b], in reference to [KV11].

This strongly contrasts with SPHFs in a group-based setting, which can handle classical ElGamal or Cramer-Shoup encryption schemes —for example [CS02, GL06]— without any modification of the decryption procedure. This is a technical hassle to carry when building on top of such an SPHF.

We therefore view as an important question to determine whether this caveat is inherent to lattice-based SPHFs, or if it can be overcome. We shall find an answer by re-introducing some harmonic analysis.

Contributions. Our main contribution consists in constructing SPHFs for standard lattice-based encryption schemes. We provide general theorems to ease the proofs of correctness and security (a.k.a., *smoothness* or *universality*) of SPHFs over standard lattice-based encryption schemes. We detail two particular instantiations: one over an IND-CCA2 encryption scheme à la Micciancio-Peikert [MP12], and one over an IND-CPA restriction of the same scheme. While the second instantiation is over a simpler language, it is a word-independent SPHF. To our knowledge, this is the first word-independent SPHF over any lattice-based language. We remark that while Zhang and Yu construct an interesting *approximate* word-independent SPHF over a lattice-based language in [ZY17], its correctness is only approximate contrary to our SPHF; and its language also relies on simulation-sound non-interactive zero-knowledge proofs, which we do not know how to construct just from lattice assumptions in the standard model.

As with many zero-knowledge-type primitives in the lattice setting [Lyu08, Lyu09] and as with the SPHFs of [KV09] and of [ZY17], there is a gap between the correctness property and the smoothness property. Concretely, smoothness holds for ciphertexts which do not decrypt to a given message, while correctness holds only for honestly generated ciphertexts. However, contrary to [KV09], we use a standard encryption scheme and do not need to tweak the decryption procedure nor the language. We thus avoid the main caveat of the latter paper.

Applications. Having built these new SPHFs, we can now proceed with several applications showing that the gap between smoothness (or universality) and correctness is not an issue in most cases. We start by proposing an efficient password-authenticated key exchange (PAKE) scheme in three flows. We do so by plugging our first SPHF in the framework from [KV09]. Following the GK-PAKE construction from [ABP15b] which is an improvement of the Groce-Katz framework [JG04, GK10], we also obtain a PAKE in two flows over lattices in the standard model. Finally, using our word-independent SPHF together with simulation-sound non-interactive zero-knowledge proofs (SS-NIZK), by following [KV11], we obtain a one-round PAKE.

Compared to the recent work of Zhang and Yu [ZY17], which proposes the first two-round lattice-based PAKE assuming in addition SS-NIZK, our two-round PAKE does not require SS-NIZK. While there exist very efficient SS-NIZKs in the random oracle model for the languages considered by Zhang and Yu, constructing SS-NIZK in the standard model under a lattice-based assumption remains an important open problem. Our two-round PAKE is thus the first two-round PAKE solely based on lattice assumptions in the standard model. In addition, our one-round PAKE assuming LWE and SS-NIZK is the first one-round PAKE in this setting and closes an open problem of [ZY17].

In addition to PAKE, we also show how to construct honest-verifier zero-knowledge proofs for any NP language from lattice-based SPHF. We conclude by showing a relaxed version of witness encryption for some lattice-based languages. Witness encryption is a very recent primitive introduced in [GGSW13] which enables a user to encrypt a message to a given word of some NP language. The message can be decrypted using a witness for the word.

Technical Overview. Let us now give a technical overview of our main contribution, namely the constructions of new lattice-based SPHFs. We focus on the language of dual-Regev ciphertexts $\boldsymbol{c}$ of 0: $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \mathbb{Z}_q^m$, where $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$ is a public matrix, while $\boldsymbol{s} \in \mathbb{Z}_q^n$ and $\boldsymbol{e} \in \mathbb{Z}_q^m$ correspond to the randomness of the ciphertext. The vector $\boldsymbol{e}$ is supposed to be small, i.e., $\boldsymbol{c}$ is close to the q-ary lattice Λ generated by $\boldsymbol{A}$.

Intuitively, an SPHF allows a prover knowing $\boldsymbol{s}$ and $\boldsymbol{e}$ to prove to a verifier that $\boldsymbol{c}$ is indeed a ciphertext of 0. The naive and natural construction works as follows.[6] The verifier generates a *small* random vector $\mathsf{hk} = \boldsymbol{h} \in \mathbb{Z}_q^m$ called a *hashing key*. It then "hashes" the ciphertext into a *hash value* $\mathsf{H} = R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle) \in \{0, 1\}$, where R is a *rounding function* from $\mathbb{Z}_q$ to $\{0, 1\}$ to be chosen later. The verifier also derives from $\mathsf{hk} = \boldsymbol{h}$, a *projection key* $\mathsf{hp} = \boldsymbol{p} = \boldsymbol{A}^t \boldsymbol{h} \in \mathbb{Z}_q^n$ that it sends to the prover. The prover can then compute the *projected hash value* $\mathsf{pH} = R(\langle \boldsymbol{p}, \boldsymbol{s} \rangle)$ from the projection key $\boldsymbol{p}$ and the randomness of the ciphertext $\boldsymbol{s}$ and $\boldsymbol{e}$. It can send this projected hash value to the verifier which will accept the proof, if pH matches its hash value H.

We remark that if indeed $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$ with $\boldsymbol{e}$ small enough (recall that $\boldsymbol{h}$ is small as well):

$$\langle \boldsymbol{h}, \boldsymbol{c} \rangle = \boldsymbol{h}^t \boldsymbol{A} \boldsymbol{s} + \boldsymbol{h}^t \boldsymbol{e} \approx \boldsymbol{h}^t \boldsymbol{A} \boldsymbol{s} = \langle \boldsymbol{p}, \boldsymbol{s} \rangle \ .$$

Hence, if R is carefully chosen, we can ensure that with high probability (e.g., at least $3/4$), $\mathsf{H} = \mathsf{pH}$, and the verifier will accept the prover's "proof." This property is called *approximate correctness*. An SPHF also needs to satisfy a security property to be useful, called *smoothness* or *universality*, which ensures that if $\boldsymbol{c}$ is far from the q-ary lattice Λ generated by $\boldsymbol{A}$ (in particular if it is an encryption of 1), then given the projection key $\boldsymbol{p}$ (and $\boldsymbol{A}$ and $\boldsymbol{c}$), the prover cannot guess the hash value H with probability more than $1/2 + \operatorname{negl}(n)$. In [KV09], Katz and Vaikuntanathan argued universality for ciphertexts $\boldsymbol{c}$, for which every multiple of $\boldsymbol{c}$ is far from the lattice Λ. To be useful in their PAKE application, the decryption procedure of the encryption scheme therefore needs to be tweaked to try to decrypt not only the ciphertext itself but also all its multiples. In particular, their construction cannot work with super-polynomial moduli.

[6] Actually, what we construct in this overview are bit-PHF and not SPHF, i.e., the hash value defined later is just a bit and the security property is universality instead of smoothness. Classical SPHFs can be derived from these bit-PHFs. See Fig. 2 and Section 2.3.

The question we wish to answer is whether universality holds without this tweak? In other words, is the condition that $j\boldsymbol{c}$ is far from Λ for all $j \neq 0$ truly necessary or is it is an artifact of the proof? To approach this question, let us discuss two case studies.

Two case studies. Let us first take a look at the special case where q is even, and where $\boldsymbol{c}$ is a perfect encryption of 1: $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s} + (0, \ldots, 0, q/2)^t$ for some $\boldsymbol{s} \in \mathbb{Z}_q^n$. We observe that

$$\langle \boldsymbol{h}, \boldsymbol{c} \rangle = \langle \boldsymbol{p}, \boldsymbol{s} \rangle + (h_m \bmod 2) \cdot q/2 \ ,$$

where h_m is the last coordinate of $\boldsymbol{h}$. In particular, the distribution of $\langle \boldsymbol{h}, \boldsymbol{c} \rangle$, when $\boldsymbol{h}$ is drawn from a discrete Gaussian (over $\mathbb{Z}^m$), conditioned on $\boldsymbol{A}$, $\boldsymbol{c}$ and $\boldsymbol{A}^t\boldsymbol{h} = \boldsymbol{p}$, is concentrated on merely 2 values out of q and is therefore far from uniform.

Yet, assuming the discrete Gaussian has large enough parameter (more precisely, twice as large as the smoothing parameter of $\mathbb{Z}$), we note that h_m is close to uniform modulo 2. In that case we observe that while $\langle \boldsymbol{h}, \boldsymbol{c} \rangle$ is not itself uniform, the rounding $R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle)$ is close to uniform when choosing the typical rounding function $R : x \in \mathbb{Z}_q \mapsto \lfloor 2x/q \rceil \mod 2$, regardless of the value of $\langle \boldsymbol{p}, \boldsymbol{s} \rangle$. So it seems that the rounding function does not only help in ensuring approximate correctness, but it can also improve universality of the scheme as well!

Unfortunately, we cannot always expect universality from this trick. Now assume that q is divisible by 3, and set $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s} + (0, \ldots, 0, q/3)^t$. This time,

$$\langle \boldsymbol{h}, \boldsymbol{c} \rangle = \langle \boldsymbol{p}, \boldsymbol{s} \rangle + (h_m \bmod 3) \cdot q/3$$

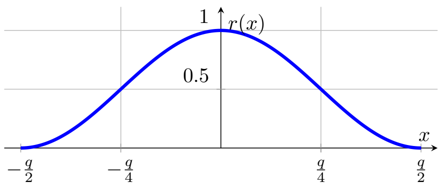
is (almost) uniformly distributed over three values, separated by $q/3$. In particular $R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle)$ will take one value with probability (roughly) 1/3, and the other value with probability (roughly) 2/3. Despite imperfect universality, this still guarantees some entropy in $\mathsf{Hash}(\boldsymbol{h}, \boldsymbol{A}, \boldsymbol{c})$ knowing $\boldsymbol{A}$, $\boldsymbol{c}$, and $\boldsymbol{p}$.

Harmonic analysis. The core of our work consists in using harmonic analysis to better understand the caveat of [KV09], namely that universality is only proven when all the multiples of the ciphertext are far from the lattice. For that, we extend the rounding function R to a q-periodic signal $\mathbb{R} \to \mathbb{R}$.

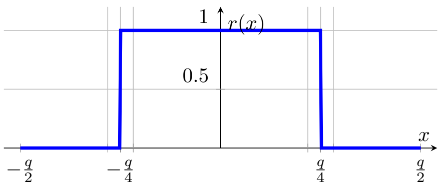
We proceed to a general analysis (Theorem 3.1), which shows that universality holds for ciphertexts $\boldsymbol{c}$ such that its multiples $j\boldsymbol{c}$ are far away from the lattice Λ, for all non-zero integers j corresponding to non-zero real harmonics of the rounding signal R.

This unravels the causes of the caveat in [KV09]: the weight of the j-th harmonic of the naive rounding function R: $x \in \mathbb{Z}_q \mapsto \lfloor 2x/q \rceil \bmod 2$ (seen as a q-periodic signal, as in Fig. 1a) is as large as $\Theta(1/j)$ for odd integers j.

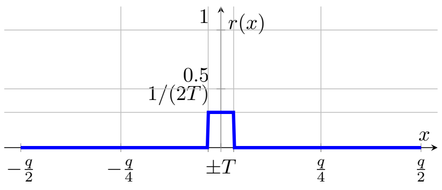
First solution (Universality, Approximate Correctness, § 3). Having identified the source of the caveat, it becomes clear how to repair it: the rounding should



(a) $r(x) = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2\pi x}{q}\right)$ (Section 3)

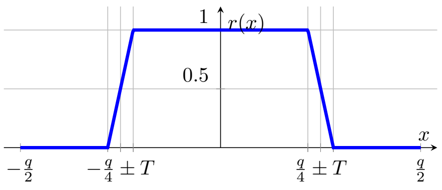


(b) $r(x) = r^{\sharp}(x) = 1 + \lfloor 2x/q \rceil \bmod 2$ (naive rounding)



(not to scale: T/q and $1/T$ are negligible)

(c) $r(x) = r^{\flat}(x)$ (used to smooth $r^{\sharp}$)



(not to scale: T/q and $1/T$ are negligible)

(d) $r(x) = (r^{\sharp} \odot r^{\flat})(x)$ (Section 4)

Fig. 1: Probability that the rounding functions $R(x)$ of Sections 3 and 4 output 1

be *randomized*, with a weight signal for which only the first harmonic is non-zero (in addition to the average), namely with a *pure cosine* weight:

$$\Pr[R(x) = 1] := \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2\pi x}{q}\right) .$$

This choice ensures universality as soon as just $1 \cdot \boldsymbol{c} = \boldsymbol{c}$ is far from the lattice Λ (Corollary 3.2 and Theorem 3.4).

This solution nevertheless only provides approximate correctness (correctness holds with probability $3/4 + o(1)$, see Lemma 3.3), which is also problematic for some applications. This can be solved using correctness amplification via error-correcting codes, but at the price of preventing the resulting SPHF to be word-independent.

Second solution (Imperfect Universality, Statistical Correctness, § 4). In our second instantiation, we therefore proceed to construct an almost-square rounding function (see Fig. 1d, $\odot$ denotes the convolution operator), which offers statistical correctness[7] and imperfect universality (namely the probability that a prover knowing only $\mathsf{hp} = \boldsymbol{p}$ can guess the hash value H is at most $1/3 + o(1)$, as proved in Theorem 4.5). This instantiation requires a more subtle analysis, taking account of *destructive interferences.*

We can then amplify universality to get statistical universality (i.e., the above probability of guessing is at most $1/2 + \mathrm{negl}(n)$ as in our first solution)

[7] More precisely, the probability of error is $\mathrm{poly}(n, \sigma)/q$, which is $\mathrm{negl}(n)$ for super-polynomial approximation factors q/σ.

while keeping a statistical correctness. Contrary to the correctness amplification, this transformation preserves the independence of the projection key from the ciphertext. In particular, if the ciphertexts are from an IND-CPA scheme such as dual-Regev, then we get the first word-independent SPHF over a lattice-based language.

We remark that our word-independent SPHF uses a *super-polynomial modulus* q, to get statistical correctness. It seems hard to construct such an SPHF for a polynomial modulus, as a word-independent SPHF for an IND-CPA encryption scheme directly yields a one-round key exchange (where each party sends a ciphertext of 0 and a projection key, and where the resulting session key is the xor of the two corresponding hash values) and we do not know of any lattice-based one-round key exchange using a polynomial modulus.

Open Question. We see as the main open question to extend our techniques to their full extent in the ring-setting. More precisely, our SPHF only produces one-bit hashes, and is easily extended to the ring-setting still asking with 1-bit hash values. This requires costly repetitions for applications, and one would hope that a ring setting variant could directly produce $\Theta(n)$-bit hash values.

Another important open question is to understand whether our techniques can further be refined to construct lattice-based IND-CCA encryption schemes without trapdoor, using ideas from the Cramer-Shoup encryption scheme [CS98, CS02] for example.

Road Map. We start by some preliminaries on lattices and SPHFs in Section 2. In particular, we define several variants of lattice-based (approximate) SPHFs (in particular universal bit-PHFs) and formally show various transformations which were only implicit in [KV09]. We also define the IND-CCA2 encryption scheme "à la Micciancio-Peikert" we will be using. In Section 3, we then show step-by-step how to construct an SPHF for IND-CCA2 ciphertexts à la Micciancio-Peikert and how to avoid the caveat of the construction of [KV09]. In Section 4, we construct a word-independent SPHF for ciphertexts under an IND-CPA scheme à la Micciancio-Peikert, when the modulus is super-polynomial. In Section 5, we conclude by exhibiting several applications.

Figure 2 summarizes our results and the paper road map. All the notions in this figure are formally defined in Section 2.

2 Preliminaries

2.1 Notations

The security parameter is denoted n. The notation $\operatorname{negl}(n)$ denotes any function f such that $f(n) = n^{-\omega(1)}$. For a probabilistic algorithm $\mathsf{alg}(\mathsf{inputs})$, we may explicit the randomness it uses with the notation $\mathsf{alg}(\mathsf{inputs}\,;\,\mathsf{coins})$, otherwise the random coins are implicitly fresh.

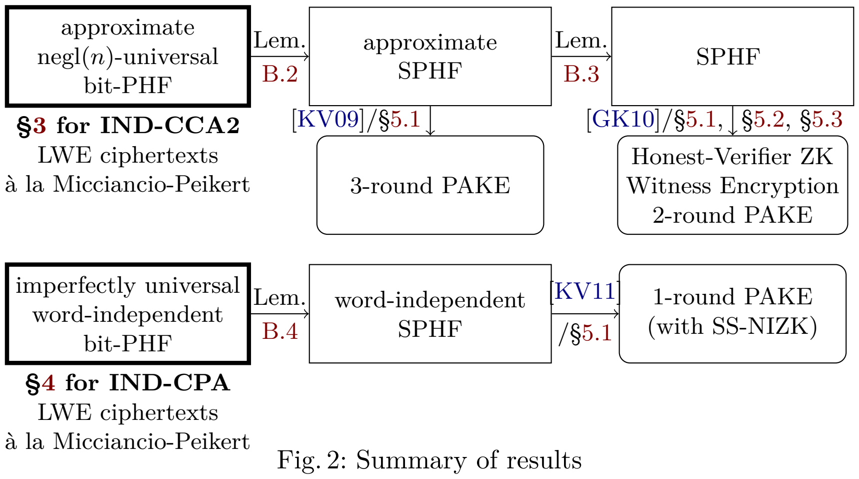


Fig. 2: Summary of results

Column vectors will be denoted by bold lower-case letters, e.g., $\boldsymbol{x}$, and matrices will be denoted by bold upper-case letters, e.g., $\boldsymbol{A}$. If $\boldsymbol{x}$ is vector and $\boldsymbol{A}$ is a matrix, $\boldsymbol{x}^t$ and $\boldsymbol{A}^t$ will denote their transpose. We use $[\boldsymbol{A}|\boldsymbol{B}]$ for the horizontal concatenation of matrices, and $[\boldsymbol{A}\,;\,\boldsymbol{B}] = [\boldsymbol{A}^t|\boldsymbol{B}^t]^t$ for the vertical concatenation. For $\boldsymbol{x} \in \mathbb{R}^m$, $\|\boldsymbol{x}\|$ will denote the canonical euclidean norm of $\boldsymbol{x}$. We will use $\mathcal{B}$ to denote the euclidean ball of radius 1, where, unless specifically stated otherwise, the ball is m-dimensional. If $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ will denote their canonical inner product, and $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$ their distance. If $E \subset \mathbb{R}^m$ is countable and discrete, we will denote $d(\boldsymbol{x}, E) = \min_{\boldsymbol{y} \in E} d(\boldsymbol{x}, \boldsymbol{y})$. For a function $f\colon E \to \mathbb{C}$ or $f\colon E \to \mathbb{R}$, $f(E)$ will denote the sum $\sum_{\boldsymbol{x} \in E} f(x)$. For $a, b \in \mathbb{R}$, $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$ will denote the closed real interval with endpoints a and b, $\lfloor a \rfloor$, $\lceil a \rceil$, and $\lfloor a \rceil$ will respectively denote the largest integer smaller than a, the smallest integer greater than a, and the closest integer to a (the largest one if there are two). The xor of two bit strings $a, b \in \{0, 1\}^k$ is denoted by $a \oplus b$. The cardinal of a finite set S is denoted $|S|$.

The modulus $q \in \mathbb{Z}$ will be taken as an odd prime, for simplicity.

2.2 Lattices and Gaussians

Lattices. An m-dimensional *lattice* Λ is a discrete subgroup of $\mathbb{R}^m$. Equivalently, Λ is a lattice if it can be written $\Lambda = \{\boldsymbol{Bs} \mid \boldsymbol{s} \in \mathbb{Z}^n\}$ where $n \leq m$, for some $\boldsymbol{B} \in \mathbb{R}^{m \times n}$, where the columns of $\boldsymbol{B}$ are linearly independent. In that case, $\boldsymbol{B}$ is called a *basis* of Λ. Then, we define the *determinant* of Λ as $\det(\Lambda) = \sqrt{\det(\boldsymbol{B}^t\boldsymbol{B})}$, which does not depend on the choice of the basis $\boldsymbol{B}$.

We define the *dual lattice* of Λ as

$$\Lambda^* = \{\boldsymbol{x} \in \mathrm{Span}_{\mathbb{R}}(\Lambda) \mid \forall \boldsymbol{y} \in \Lambda, \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}\} \ .$$

Recall the identity $(\varLambda^*)^* = \varLambda$. Given $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ where $m \geq n$, and modulus $q \geq 2$, we define the following q-ary lattices

$$\varLambda(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{s} \mid \boldsymbol{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m \quad , \quad \varLambda^\perp(\boldsymbol{A}) = \{\boldsymbol{h} \in \mathbb{Z}^m \mid \boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{0}^t \bmod q\} \ .$$

Note that up to a scaling factor, $\varLambda(\boldsymbol{A})$ and $\varLambda^\perp(\boldsymbol{A})$ are dual of each other: $\varLambda(\boldsymbol{A}) = q \cdot \varLambda^\perp(\boldsymbol{A})^*$. For a syndrome $\boldsymbol{p} \in \mathbb{Z}_q^n$, we define the coset of $\varLambda^\perp(\boldsymbol{A})$:

$$\varLambda^\perp_{\boldsymbol{p}}(\boldsymbol{A}) = \{\boldsymbol{h} \in \mathbb{Z}^m \mid \boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{p}^t \bmod q\} \ .$$

When there is no confusion about which matrix $\boldsymbol{A}$ is used, we will simply denote these lattices $\varLambda$, $\varLambda^\perp$, and $\varLambda^\perp_{\boldsymbol{p}}$ respectively.

Gaussians. If $s > 0$ and $\boldsymbol{c} \in \mathbb{R}^m$, we define the *Gaussian weight function* on $\mathbb{R}^m$ as

$$\rho_{s,\boldsymbol{c}} \colon \boldsymbol{x} \mapsto \exp(-\pi\|\boldsymbol{x} - \boldsymbol{c}\|^2/s^2).$$

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

$$\forall \boldsymbol{x} \in \varLambda,\ D_{\varLambda,s,\boldsymbol{c}}(\boldsymbol{x}) = \frac{\rho_{s,\boldsymbol{c}}(x)}{\rho_{s,\boldsymbol{c}}(\varLambda)} \ .$$

When $\boldsymbol{c} = \boldsymbol{0}$, we will simply write ρ_s and $D_{\varLambda,s}$. We recall the tail-bound of Banaszczyk for discrete Gaussians:

Lemma 2.1 ([Ban93, Lemma 1.5], as stated in [MR04, Lemma 2.10]). *For any $c > 1/\sqrt{2\pi}$, m-dimensional lattice $\varLambda$ and any vector $\boldsymbol{v} \in \mathbb{R}^m$:*

$$\rho_s(\varLambda \setminus sc\sqrt{m}\mathcal{B}) \leq C^m\rho_s(\varLambda) \quad , \qquad \rho_s((\varLambda + \boldsymbol{v}) \setminus sc\sqrt{m}\mathcal{B}) \leq 2C^m\rho_s(\varLambda) \ .$$

where $C = c\sqrt{2\pi e} \cdot e^{-\pi c^2} < 1$.

An important quantity associated to a lattice is its *smoothing parameter*, introduced by Micciancio and Regev [MR04]:

Definition 2.2 (Smoothing parameter [MR04]). *For $\epsilon > 0$, the smoothing parameter of a lattice $\varLambda$, denoted $\eta_\epsilon(\varLambda)$, is the smallest $s > 0$ such that $\rho_{1/s}(\varLambda^* \setminus \{0\}) \leq \epsilon$.*

The following lemma states that if the parameter of the discrete Gaussian is above the smoothing parameter of the lattice, then the Gaussian weight of the cosets of $\varLambda$ are essentially the same:

Lemma 2.3 ([Reg05, Claim 3.8]). *For any lattice $\varLambda \subset \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^m$, and $s \geq \eta_\epsilon(\varLambda)$:*

$$(1-\epsilon)s^m \det(\varLambda^*) \leq \rho_s(\varLambda + \boldsymbol{c}) \leq (1+\epsilon)s^m \det(\varLambda^*) \ .$$

The smoothing parameter of the dual of a random q-ary lattice can be controlled using the following:

Lemma 2.4 (Corollary of [MP12, Lemma 2.4]). *Fix parameters n, q a prime, and $m \geq \Theta(n \log q)$. Let $\epsilon \geq 2^{-O(n)}$ and $s > 2\eta_\epsilon(\mathbb{Z}^m)$. Fix $0 < \delta \leq 1$. Then, for $\boldsymbol{A}$ uniformly random in $\mathbb{Z}_q^{m \times n}$, we have $s \geq \eta_{2\epsilon/\delta}(\Lambda^\perp(\boldsymbol{A}))$ except with probability at most δ over the choice of $\boldsymbol{A}$.*

To instantiate the above, we recall the smoothing parameter of $\mathbb{Z}^m$.

Lemma 2.5 (Corollary of [MR04, Lemma 3.3]). *For all integer $m \geq 1$, $\epsilon \in (0, 1/2)$, the smoothing parameter of $\mathbb{Z}^m$ satisfies $\eta_\epsilon(\mathbb{Z}^m) \leq C\sqrt{\log(m/\epsilon)}$ for some universal constant $C > 0$.*

Harmonic analysis. Let us recall the exponential basis of periodic functions and their vectorial analogues:

$$e_x : y \mapsto \exp(2i\pi xy) \ , \qquad\qquad e_{\boldsymbol{x}} : \boldsymbol{y} \mapsto \exp(2i\pi\langle \boldsymbol{x}, \boldsymbol{y}\rangle) \ .$$

The Fourier transform of $f : \mathbb{R}^m \to \mathbb{C}$ is defined by:

$$\hat{f}(\boldsymbol{\xi}) = \int_{\mathbb{R}^m} f(\boldsymbol{x}) e^{-2i\pi\langle \boldsymbol{x}, \boldsymbol{\xi}\rangle} d\boldsymbol{x} \ .$$

The Fourier transform of the Gaussian weight function ρ_s is $\widehat{\rho_s} = s^m \rho_{1/s}$. Recall the time-shift-phase-shift identity: if $g(\boldsymbol{x}) = f(\boldsymbol{x}) e_{\boldsymbol{z}}(\boldsymbol{x})$ for some $\boldsymbol{z} \in \mathbb{R}^m$, then $\hat{g}(\boldsymbol{\xi}) = \hat{f}(\boldsymbol{\xi} - \boldsymbol{z})$. Similarly, if $g(\boldsymbol{x}) = f(\boldsymbol{x} + \boldsymbol{t})$ for some $\boldsymbol{t} \in \mathbb{R}^m$, then $\hat{g}(\boldsymbol{\xi}) = \hat{f}(\boldsymbol{\xi}) e_{\boldsymbol{t}}(\boldsymbol{\xi})$. For two functions $f, g : \mathbb{R}^m \to \mathbb{C}$, we will denote by $f \odot g$ their convolution product:

$$f \odot g(\boldsymbol{x}) = \int_{\mathbb{R}^m} f(\boldsymbol{y}) g(\boldsymbol{x} - \boldsymbol{y}) d\boldsymbol{y} \ .$$

The Fourier transform turns convolutions into pointwise products, and conversely:

$$\widehat{f \odot g}(\boldsymbol{\xi}) = \hat{f}(\boldsymbol{\xi}) \cdot \hat{g}(\boldsymbol{\xi}) \ , \qquad\qquad \widehat{f \cdot g}(\boldsymbol{\xi}) = \hat{f}(\boldsymbol{\xi}) \odot \hat{g}(\boldsymbol{\xi}) \ .$$

Finally, let us recall the Poisson summation formula:

Lemma 2.6 (Poisson summation formula). *For any lattice Λ and $f : \mathbb{R}^m \to \mathbb{C}$, we have $f(\Lambda) = \det(\Lambda^*) \hat{f}(\Lambda^*)$.*

Learning with Errors.

Definition 2.7 (Learning with Errors (LWE)). *Let $q \geq 2$, and χ be a distribution over $\mathbb{Z}$. The Learning with Errors problem $\mathrm{LWE}_{\chi,q}$ consists in, given polynomially many samples, distinguishing the two following distributions:*

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

In [Reg05], Regev showed that for $\chi = D_{\mathbb{Z},\sigma}$, for any $\sigma \geq 2\sqrt{n}$, and q such that $q/\sigma = \mathrm{poly}(n)$, $\mathrm{LWE}_{\chi,q}$ is at least as hard as solving worst-case SIVP for polynomial approximation factors.

Trapdoor for LWE. Throughout this paper, we will use the trapdoors introduced in [MP12] to build our public matrix $\boldsymbol{A}$. Define $g_{\boldsymbol{A}}(\boldsymbol{s}, \boldsymbol{e}) = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$, let $\boldsymbol{G}^t = \boldsymbol{I}_n \otimes \boldsymbol{g}^t$, where $\boldsymbol{g}^t = [1, 2, \ldots, 2^k]$ and $k = \lceil \log q \rceil - 1$, and let $\boldsymbol{H} \in \mathbb{Z}_q^{n \times n}$ be invertible.

Lemma 2.8 ([MP12, Theorems 5.1 and 5.4]). *There exist two PPT algorithms* $\mathsf{TrapGen}$ *and* $g^{-1}_{(\cdot)}$ *with the following properties assuming* $q \geq 2$ *and* $m \geq \Theta(n \log q)$*:*

- $\mathsf{TrapGen}(1^n, 1^m, q)$ *outputs* $(\boldsymbol{T}, \boldsymbol{A}_0)$*, where the distribution of the matrix* $\boldsymbol{A}_0$ *is at negligible statistical distance from uniform in* $\mathbb{Z}_q^{m \times n}$*, and such that* $\boldsymbol{T}\boldsymbol{A}_0 = \boldsymbol{0}$*, where* $s_1(\boldsymbol{T}) \leq O(\sqrt{m})$ *and where* $s_1(\boldsymbol{T})$ *is the operator norm of* $\boldsymbol{T}$*, which is defined as* $\max_{\boldsymbol{x} \neq 0} \|\boldsymbol{T}\boldsymbol{x}\| / \|\boldsymbol{x}\|$.[8]
- *Let* $(\boldsymbol{T}, \boldsymbol{A}_0) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$*. Let* $\boldsymbol{A}_{\boldsymbol{H}} = \boldsymbol{A}_0 + [\boldsymbol{0}\,; \boldsymbol{G}\boldsymbol{H}]$ *for some invertible matrix* $\boldsymbol{H}$ *called a* tag*. Then, we have* $\boldsymbol{T}\boldsymbol{A}_{\boldsymbol{H}} = \boldsymbol{G}\boldsymbol{H}$*. Furthermore, if* $\boldsymbol{x} \in \mathbb{Z}_q^m$ *can be written as* $\boldsymbol{A}_{\boldsymbol{H}}\boldsymbol{s} + \boldsymbol{e}$ *where* $\|\boldsymbol{e}\| \leq B' := q/\Theta(\sqrt{m})$*, then* $g^{-1}_{\boldsymbol{A}_{\boldsymbol{H}}}(\boldsymbol{T}, \boldsymbol{x}, \boldsymbol{H})$ *outputs* $(\boldsymbol{s}, \boldsymbol{e})$*.*

More precisely, to sample $(\boldsymbol{T}, \boldsymbol{A}_0)$ with $\mathsf{TrapGen}$, we sample a uniform $\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{\bar{m} \times n}$ where $\bar{m} = m - nk = \Theta(n \log q)$, and some $\boldsymbol{R} \leftarrow \mathcal{D}^{nk \times \bar{m}}$, where the distribution $\mathcal{D}^{nk \times \bar{m}}$ assigns probability $1/2$ to 0, and $1/4$ to ± 1. We output $\boldsymbol{T} = [-\boldsymbol{R} \,|\, \boldsymbol{I}_{nk}]$ along with $\boldsymbol{A}_0 = [\bar{\boldsymbol{A}}\,; \boldsymbol{R}\bar{\boldsymbol{A}}]$. Then, given a tag $\boldsymbol{H}$, we have: $\boldsymbol{T}(\boldsymbol{A}_0 + [\boldsymbol{0}\,; \boldsymbol{G}\boldsymbol{H}]) = \boldsymbol{G}\boldsymbol{H}$.

Tag-IND-CCA2 LWE encryption à la Micciancio-Peikert. For our applications, we will need a (labelled) encryption scheme that is IND-CCA2 (the definition is given in Appendix A.1). This can be built generically and efficiently from a tag-IND-CCA2 encryption scheme, as recalled in Appendix A.2. Below, we describe a simplified variant of the scheme of [MP12, Sec. 6.3].

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

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

Let $(\boldsymbol{T}, \boldsymbol{A}_0) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$. The public encryption key is $\mathsf{ek} = \boldsymbol{A}_0$, and the secret decryption key is $\mathsf{dk} = \boldsymbol{T}$.

- $\mathsf{Encrypt}(\mathsf{ek} = \boldsymbol{A}_0,\ u \in \mathcal{U},\ \mu \in \{0,1\})$ encrypts the message μ under the public key ek and for the tag u, as follows: Let $\boldsymbol{A}_u = \boldsymbol{A}_0 + [\boldsymbol{0}\,; \boldsymbol{G}h(u)]$. Pick

[8] The bound on $s_1(\boldsymbol{T})$ holds except with probability at most 2^{-n} in the original construction, but for convenience we assume the algorithm restarts if it does not hold.

$\boldsymbol{s} \in \mathbb{Z}_q^n$, $\boldsymbol{e} \leftarrow D_{\mathbb{Z},t}^m$ where $t = \sigma\sqrt{m} \cdot \omega(\sqrt{\log n})$. Restart if $\|\boldsymbol{e}\| > B$, where $B \coloneqq 2t\sqrt{m}$.[9] Output the ciphertext:

$$\boldsymbol{c} = \boldsymbol{A}_u \boldsymbol{s} + \boldsymbol{e} + \mathsf{Encode}(\mu) \bmod q \ .$$

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

$$\begin{cases} \mu & \text{if } g_{\boldsymbol{A}_u}^{-1}(\boldsymbol{T}, 2\boldsymbol{c}, h(u)) = 2\boldsymbol{e} + (0, \ldots, 0, \mu) \text{ where } \boldsymbol{e} \in \mathbb{Z}^m \text{ and } \|\boldsymbol{e}\| \le B' \ , \\ \perp & \text{otherwise.}^{10} \end{cases}$$

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

$$\mu' \coloneqq \mathsf{Decrypt}(\boldsymbol{T}, u, \boldsymbol{c}) \neq \perp \quad \Longleftrightarrow \quad d(\boldsymbol{c} - \mathsf{Encode}(\mu'), \Lambda(\boldsymbol{A}_u)) < B' \ .$$

Suppose that $m \ge \Theta(n \log q)$. Note that $d(\mathsf{Encode}(1), \Lambda(\boldsymbol{A}_u)) > B'$ simultaneously for all u with overwhelming probability over the randomness of $\mathsf{TrapGen}$ (using a union bound, as in [GPV08, Lemma 5.3] for instance). Then, by Lemma 2.8, the scheme is correct as long as $B \le B'$, or equivalently $\sigma m^{3/2} \cdot \omega(\sqrt{\log n}) \le q$.

Theorem 2.9. *Assume $m \ge \Theta(n \log q)$. The above scheme is tag-IND-CCA2 assuming the hardness of the LWE$_{\chi,q}$ problem for $\chi = D_{\mathbb{Z},\sigma}$.*

The precise definition for tag-IND-CCA2 is detailed in Appendix A.1, and the proof is given in Appendix A.3.

Remark 2.10. If a constant tag u is hardcoded in $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$, then the resulting encryption scheme is just an IND-CPA scheme using trapdoors from [MP12].

Lemma 2.11. *Assume $m \ge \Theta(n \log q)$. With $\boldsymbol{A}_0$ sampled as above, except with probability 2^{-n}, it holds that for all $u \in \mathcal{U}$, $\eta_{2^{-n}}(\Lambda^\perp(\boldsymbol{A}_u)) \le C\sqrt{n}$ for some universal constant C.*

Proof. Note that $\boldsymbol{A}_0$ is (about) uniform under the randomness of $\mathsf{TrapGen}$, and so is $\boldsymbol{A}_u$ for a fixed $u \in \mathcal{U}$. Apply Lemma 2.4 and Lemma 2.5 with $\epsilon = 8^{-n}/2$ and $\delta = 4^{-n}$ to $\boldsymbol{A}_u$, ensuring that $\eta_{2^{-n}}(\Lambda^\perp(\boldsymbol{A}_u)) \le C\sqrt{n}$ except with probability δ. Conclude by the union bound over the 2^n elements $u \in \mathcal{U}$. $\square$

2.3 Approximate Smooth Projective Hash Functions

We consider approximate smooth projective hash functions (approximate SPHFs) defined in [KV09].

[9] This happens only with exponentially small probability $2^{-\Theta(n)}$ by Lemma 2.1.

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

Languages. We consider a family of languages $(\mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}})_{\mathsf{lpar},\mathsf{ltrap}}$ indexed by some *parameter* lpar and some *trapdoor* ltrap, together with a family of NP languages $(\widetilde{\mathscr{L}}_{\mathsf{lpar}})_{\mathsf{lpar}}$ indexed by some parameter lpar, with witness relation $\widetilde{\mathscr{R}}_{\mathsf{lpar}}$, such that:

$$\widetilde{\mathscr{L}}_{\mathsf{lpar}} = \{\chi \in \mathcal{X}_{\mathsf{lpar}} \mid \exists w,\ \widetilde{\mathscr{R}}_{\mathsf{lpar}}(\chi, w) = 1\} \ \subseteq\ \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}} \ \subseteq\ \mathcal{X}_{\mathsf{lpar}}\ ,$$

where $(\mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar}}$ is a family of sets. The trapdoor ltrap and the parameter lpar are generated by a polynomial-time algorithm $\mathsf{Setup.lpar}$ which takes as input a unary representation of the security parameter n. We suppose that membership in $\mathcal{X}_{\mathsf{lpar}}$ and $\widetilde{\mathscr{R}}_{\mathsf{lpar}}$ can be checked in polynomial time given lpar and that membership in $\mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}}$ can be checked in polynomial time given lpar and ltrap. The parameters lpar and ltrap are often omitted when they are clear from context.

We are mostly interested in languages of ciphertexts.

Example 2.12 (Languages of Ciphertexts). Let $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a labeled encryption scheme. We define the following languages ($\mathsf{Setup.lpar} = \mathsf{KeyGen}$ and $(\mathsf{ltrap}, \mathsf{lpar}) = (\mathsf{dk}, \mathsf{ek})$):

$$\begin{aligned}\widetilde{\mathscr{L}} &= \{(\mathsf{label}, C, M) \mid \exists \rho,\ C = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{label}, M; \rho)\}\ ,\\ \mathscr{L} &= \{(\mathsf{label}, C, M) \mid \mathsf{Decrypt}(\mathsf{dk}, \mathsf{label}, C) = M\}\ ,\end{aligned}$$

where the witness relation $\widetilde{\mathscr{R}}$ is implicitly defined as: $\widetilde{\mathscr{R}}((\mathsf{label}, C, M), \rho) = 1$ if and only if $C = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{label}, M; \rho)$.

Approximate SPHFs. Let us now define approximate SPHFs following [KV09].

Definition 2.13. *Let $(\widetilde{\mathscr{L}}_{\mathsf{lpar}} \subseteq \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}} \subseteq \mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar},\mathsf{ltrap}}$ be languages defined as above. An approximate smooth projective hash function (SPHF) for these languages is defined by four probabilistic polynomial-time algorithms:*

- $\mathsf{HashKG}(\mathsf{lpar})$ *generates a hashing key* hk *for the language parameter* lpar*;*
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *derives a projection key* hp *from the hashing key* hk*, the language parameter* lpar*, and the word* χ*;*
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs a hash value* $\mathsf{H} \in \{0,1\}^{\nu}$ *(for some positive integer* $\nu = \Omega(n)$*) from the hashing key* hk*, for the word* $\chi \in \mathcal{X}_{\mathsf{lpar}}$ *and the language parameter* lpar*;*
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *outputs a projected hash value* $\mathsf{pH} \in \{0,1\}^{\nu}$ *from the projection key* hp*, and the witness* w*, for the word* $\chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar}}$ *(i.e.,* $\widetilde{\mathscr{R}}_{\mathsf{lpar}}(\chi, w) = 1$*) and the language parameter* lpar*;*

which satisfy the following properties:

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

$$\Pr_{\mathsf{hk}}\left[\mathsf{HW}(\mathsf{Hash}(\mathsf{hk},\mathsf{lpar},\chi),\mathsf{ProjHash}(\mathsf{hp},\mathsf{lpar},\chi,w)) > \epsilon \cdot \nu\right] = \operatorname{negl}(n) \ ,$$

where the probability is taken over the choice of $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar})$ *and the random coins of* Hash *and* $\mathsf{ProjHash}$.[11]

– ***Smoothness.*** *For any* $n \in \mathbb{N}$*, if* $(\mathsf{ltrap},\mathsf{lpar}) \leftarrow \mathsf{Setup.lpar}(1^n)$*, with overwhelming probability over the randomness of* $\mathsf{Setup.lpar}$*, for all* $\chi \in \mathcal{X} \setminus \mathscr{L}_{\mathsf{lpar}}$ *the following distributions have statistical distance negligible in* n*:*

$$\left\{(\mathsf{lpar},\chi,\mathsf{hp},\mathsf{H}) \;\middle|\; \begin{array}{l}\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar}),\ \mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk},\mathsf{lpar},\chi),\\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk},\mathsf{lpar},\chi)\end{array}\right\} \ ,$$
$$\left\{(\mathsf{lpar},\chi,\mathsf{hp},\mathsf{H}) \;\middle|\; \begin{array}{l}\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar}),\ \mathsf{H} \leftarrow \{0,1\}^\nu,\\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk},\mathsf{lpar},\chi)\end{array}\right\} \ .$$

Finally, an approximate SPHF is called an SPHF if it is 0-correct. In that case, we also say that the SPHF is *statistically correct*.

Approximate Word-Independent SPHFs. For some applications, in particular the one-round PAKE from [KV11], a stronger notion of SPHF is required, where the projection key hp does not depend on the word χ and the smoothness holds even if the word is chosen adaptively after seeing the projection key. We call such SPHFs approximate word-independent SPHFs and we formally define them in Appendix B.1.

Approximate universal bit-PHFs. Instead of directly building (approximate) (word-independent) SPHF, we actually build what we call (approximate) (word-independent) universal bit-PHF.

Definition 2.14. *An approximate universal bit projective hash function (bit-PHF) is defined as in Definition 2.13 except that the hash values are bits* ($\nu = 1$)*, and that approximate correctness and smoothness are replaced by the following properties:*

– ***Approximate correctness.*** *The bit-PHF is* ϵ*-correct if for any* $n \in \mathbb{N}$*, if* $(\mathsf{ltrap},\mathsf{lpar}) \leftarrow \mathsf{Setup.lpar}(1^n)$*, with overwhelming probability over the randomness of* $\mathsf{Setup.lpar}$*, for any* $\chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar},\mathsf{ltrap}}$*:*

$$\Pr_{\mathsf{hk}}\left[\mathsf{Hash}(\mathsf{hk},\mathsf{lpar},\chi) = \mathsf{ProjHash}(\mathsf{hp},\mathsf{lpar},\chi,w)\right] \geq 1 - \epsilon \ ,$$

where the probability is taken over the choice of $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar})$ *and the random coins of* Hash *and* $\mathsf{ProjHash}$.

[11] Contrary to previously known SPHFs, some of our SPHFs have randomized algorithms Hash and $\mathsf{ProjHash}$.

– ***Universality.***[12] *The bit-PHF is ϵ-universal if, for any $n \in \mathbb{N}$, if* $(\mathsf{ltrap}, \mathsf{lpar}) \leftarrow \mathsf{Setup.lpar}(1^n)$*, with overwhelming probability over the randomness of* $\mathsf{Setup.lpar}$*, for any word $\chi \in \mathcal{X} \setminus \mathscr{L}_{\mathsf{lpar}}$, any projection key* hp*:*

$$\left| 2 \cdot \Pr_{\mathsf{hk}}\left[\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi) = 1 \mid \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)\right] - 1 \right| \leq \epsilon \ ,$$

where the probability is taken over the choice of $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar})$ *and the random coins of* Hash*. The bit-PHF is said to be* statistically universal *if it is* $\mathrm{negl}(n)$*-universal. Otherwise, the bit-PHF is said to be* imperfectly universal.

An approximate bit-PHF is called a bit-PHF if it is $\mathrm{negl}(n)$-correct. In that case, the bit-PHF is said to be *statistically correct.* Furthermore, an (approximate) bit-PHF is called an (approximate) (word-independent) bit-PHF, if hp does not depend on the word χ.

From Bit-PHFs to SPHFs. In Appendix B.2, we show how to generically convert an approximate ϵ-correct $\mathrm{negl}(n)$-universal bit-PHF into an approximate $(\epsilon + \epsilon')$-correct SPHF (for any positive constant ϵ') and then into an SPHF. This is used in our first construction in Section 3. These transformations were implicit in [KV09]. We should point out that even if the original bit-PHF was word-independent, the resulting (approximate) SPHF would still not be word-independent: its projection key depends on the word χ. If there was way to avoid this restriction, we actually would get the first one-round key exchange based on LWE with polynomial modulus.

In Appendix B.2, we also show how to generically convert an ϵ-universal *word-independent* bit-PHF into a *word-independent* SPHF, by amplifying the smoothness or universality property (assuming $1 - \epsilon \geq 1/\mathrm{poly}(n)$). We should point out that the original word-independent bit-PHF is supposed to be statistically correct, contrary to the previous transformation where it could just be approximately correct.

We recall that the above transformations were summarized in Fig. 2 together with our results.

3 SPHF for IND-CCA2 Ciphertexts

As we have shown in Section 2.3, there exists a generic transformation from approximate bit-PHF to a regular approximate SPHF or even classical SPHF. So, in this section, we are going to focus on building such an approximate bit-PHF. For the sake of simplicity, in this section we often call such an approximate bit-PHF simply a bit-PHF.

[12] Our definition of universality is equivalent to the one of Cramer and Shoup in [CS02], up to the use of language parameters.

3.1 Languages and Natural Bit-PHF

Languages. We want to construct an (approximate) bit-PHF for the language of ciphertexts (Example 2.12) for our IND-CCA2 LWE encryption à la Micciancio-Peikert described in Section 2.2. More generally our approach works with typical trapdoored LWE encryption schemes [GPV08, CHKP10].

We first remark that it is sufficient to construct a bit-PHF for the tag-IND-CCA2 version, i.e., for the following languages:

$$\begin{aligned}\widetilde{\mathscr{L}} &= \{(u, \boldsymbol{c}, \mu) \mid \exists \boldsymbol{s}, \boldsymbol{e},\ \boldsymbol{c} \leftarrow \mathsf{Encrypt}(\boldsymbol{A}_0, u, \mu; \boldsymbol{s}, \boldsymbol{e})\} \\ &\subseteq \{(u, \boldsymbol{c}, \mu) \mid d(\boldsymbol{c} - \mathsf{Encode}(\mu),\ \Lambda(\boldsymbol{A}_u)) \le B\}\ , \\ \mathscr{L} &= \{(u, \boldsymbol{c}, \mu) \mid \mathsf{Decrypt}(\boldsymbol{T}, u, \boldsymbol{c}) = \mu\} \\ &= \{(u, \boldsymbol{c}, \mu) \mid d(\boldsymbol{c} - \mathsf{Encode}(\mu),\ \Lambda(\boldsymbol{A}_u)) \le B'\}\ ,\end{aligned}$$

where $u \in \mathcal{U}$, $\boldsymbol{c} \in \mathbb{Z}_q^m$, $\mu \in \{0,1\}$, $(\mathsf{ltrap}, \mathsf{lpar}) = (\boldsymbol{T}, \boldsymbol{A}_0) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q) = \mathsf{Setup.lpar}(1^n)$, and where $\mathsf{Encrypt}$, $\mathsf{Decrypt}$, B, and B' are defined in Section 2.2. Indeed, the signature parts, used to transform the tag-IND-CCA2 encryption scheme into a labeled IND-CCA2 encryption scheme (see Appendix A.2), can be publicly checked by anyone, therefore one can generically adapt the bit-PHF by overriding Hash to a fresh uniform random value when the signature is invalid.

We can now fix the tag $u \in \mathcal{U}$ for the rest of this section, and will simply denote $\boldsymbol{A}$ for $\boldsymbol{A}_u$ and Λ for $\Lambda(\boldsymbol{A}_u)$. Also, note that $(u, \boldsymbol{c}, 1) \in \widetilde{\mathscr{L}}$ (resp. $\mathscr{L}$) is equivalent to $(u, \boldsymbol{c} - \mathsf{Encode}(1), 0) \in \widetilde{\mathscr{L}}$ (resp $\mathscr{L}$). Therefore we can focus only on the languages of ciphertexts of 0 for a fixed tag u, and we restrict our languages to

$$\begin{aligned}\widetilde{\mathscr{L}} &= \{\boldsymbol{c} \in \mathbb{Z}_q^m \mid \exists \boldsymbol{s}, \boldsymbol{e},\ \boldsymbol{c} \leftarrow \mathsf{Encrypt}(\boldsymbol{A_0}, 0, u; \boldsymbol{s}, \boldsymbol{e})\} \subseteq \{\boldsymbol{c} \in \mathbb{Z}_q^m \mid d(\boldsymbol{c}, \Lambda) \le B\}\ , \\ \mathscr{L} &= \{\boldsymbol{c} \in \mathbb{Z}_q^m \mid \mathsf{Decrypt}(\boldsymbol{T}, \boldsymbol{c}, u) = 0\} \qquad\qquad = \{\boldsymbol{c} \in \mathbb{Z}_q^m \mid d(\boldsymbol{c}, \Lambda) \le B'\}\ ,\end{aligned}$$

for the rest of this section.

Natural Bit-PHF. A natural approach to define an approximate bit-PHF is the following:

– $\mathsf{HashKG}(\boldsymbol{A})$ outputs $\mathsf{hk} = \boldsymbol{h} \leftarrow D_{\mathbb{Z},s}^m$;
– $\mathsf{ProjKG}(\boldsymbol{h}, \boldsymbol{A})$ outputs $\mathsf{hp} = \boldsymbol{p} = \boldsymbol{A}^t\boldsymbol{h}$;
– $\mathsf{Hash}(\boldsymbol{h}, \boldsymbol{A}, \boldsymbol{c})$ outputs $\mathsf{H} = R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle)$;
– $\mathsf{ProjHash}(\boldsymbol{p}, \boldsymbol{A}, \boldsymbol{c}, (\boldsymbol{s}, \boldsymbol{e}))$ outputs $\mathsf{pH} = R(\langle \boldsymbol{p}, \boldsymbol{s} \rangle)$;

where R is a *rounding* function to be chosen later and $s > 0$ is a parameter to be chosen later too.

3.2 Universality

Naive approach. For now let us just assume $R : \mathbb{Z}_q \to \mathbb{Z}_2$ to be the usual rounding function $R(x) = \lfloor 2x/q \rceil \bmod 2$, as in [KV09]. We have:

$$\langle \boldsymbol{h}, \boldsymbol{c} \rangle = \boldsymbol{h}^t(\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) = \langle \boldsymbol{p}, \boldsymbol{s} \rangle + \langle \boldsymbol{h}, \boldsymbol{e} \rangle \approx \langle \boldsymbol{p}, \boldsymbol{s} \rangle\ ,$$

which guarantees correctness whenever $c \in \widetilde{\mathscr{L}}$. Indeed $\langle \boldsymbol{h}, \boldsymbol{c} \rangle$ is almost uniform for large enough parameter s, therefore $R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle) = R(\langle \boldsymbol{p}, \boldsymbol{s} \rangle)$ will hold except with probability $\approx 2|\langle \boldsymbol{h}, \boldsymbol{e} \rangle|/q$.

For universality, we need to prove that $\mathsf{Hash}(\boldsymbol{h}, \boldsymbol{A}, \boldsymbol{c}) = \langle \boldsymbol{h}, \boldsymbol{c} \rangle$ is uniform given the knowledge of $\boldsymbol{A}, \boldsymbol{p}$ and $\boldsymbol{c}$, when $\boldsymbol{c} \notin \mathscr{L}$. Unfortunately, this seems to require a stronger assumption than $\boldsymbol{c} \notin \mathscr{L}$, more precisely, that $j \cdot \boldsymbol{c} \notin \mathscr{L}$ for all $j \in \mathbb{Z}_q^*$: this is the key lemma [GPV08, Lemma 5.3] (from [KV09, Lemma 2]).

The caveat is that it is necessary not only for $\boldsymbol{c}$ to be far from Λ, but also for all its non-zero multiples modulo q: the language is extended to $\mathscr{L}' = \{\boldsymbol{c} \mid \exists j \in \mathbb{Z}_q^*, j\boldsymbol{c} \in \mathscr{L}\}$. Algorithmically, the price to pay is that the decryption function must be changed, and that the usual LWE decryption now must be attempted for each multiple $j\boldsymbol{c}$ of $\boldsymbol{c}$ to ensure universality for words outside $\mathscr{L}'$. This makes the new decryption very inefficient since q is typically quite a large $\mathrm{poly}(n)$. This change of language is also a technical hassle for constructing protocols above the bit-PHF (or the resulting SPHF).

Note that the key lemma ensures uniformity of $\langle \boldsymbol{h}, \boldsymbol{c} \rangle$, while we only need the uniformity of $R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle)$. We show in the technical overview of the introduction that this condition is truly necessary and is not an artifact of the proof, at least for $j = 3$ by considering $\boldsymbol{c} = \boldsymbol{As} + (0, \ldots, 0, q/3)^t$ (with q assumed to be divisible by 3 for the sake of simplicity).

But what should happen in more general cases?

Harmonic analysis. Let us fix $\boldsymbol{p} \in \mathbb{Z}_q^n$ and $\boldsymbol{c} \in \mathbb{Z}_q^m$. For the rest of the section, we restrict the rounding function R to have binary values $\{0, 1\}$, yet this function may be probabilistic.

We want to study the conditional probability $P = \Pr[R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle) = 1 \mid \boldsymbol{h}^t \boldsymbol{A} = \boldsymbol{p}^t]$, where the probability is taken over the randomness of R and the distribution of $\boldsymbol{h}$ (conditioned on $\boldsymbol{h}^t \boldsymbol{A} = \boldsymbol{p}^t$); we want P to be not too far from $1/2$ when $\boldsymbol{c} \notin \mathscr{L}$. For $x \in \mathbb{Z}$, denote by $r(x)$ the probability that $R(x \bmod q) = 1$. Because $r : \mathbb{Z} \to [0, 1]$ is q-periodic, it can be interpolated over the reals by a function of the form:

$$r = \sum_{j \in \mathbb{Z}_q} \hat{r}_j \cdot e_{j/q} \quad ,$$

where the complex values $\hat{r}_j \in \mathbb{C}$ are the Fourier coefficients of $r \colon \mathbb{Z} \to [0, 1]$. Note that as we are only interested in the restriction of r on $\mathbb{Z}$ (which is q-periodic), we only need q harmonics to fully describe r. Also note that $r(x) \in [0, 1]$ for all $x \in \mathbb{Z}_q$, so that $|\hat{r}_j| \leq 1$ for all j.

We rewrite:

$$P = \sum_{\boldsymbol{h} \in \Lambda_{\boldsymbol{p}}^{\perp}} \frac{\rho_s(\boldsymbol{h})}{\rho_s(\Lambda_{\boldsymbol{p}}^{\perp})} \cdot r(\langle \boldsymbol{h}, \boldsymbol{c} \rangle) = \frac{1}{\rho_s(\Lambda_{\boldsymbol{p}}^{\perp})} \sum_{j \in \mathbb{Z}_q} \hat{r}_j \sum_{\boldsymbol{h} \in \Lambda^{\perp}} (\rho_s \cdot e_{j\boldsymbol{c}/q})(\boldsymbol{h} + \boldsymbol{h}_0) \quad ,$$

where $\boldsymbol{h}_0$ is any vector of the coset $\Lambda_{\boldsymbol{p}}^{\perp}$. We will now apply the Poisson Summation Formula (Lemma 2.6): $f(\Lambda^{\perp}) = \det((\Lambda^{\perp})^*) \hat{f}((\Lambda^{\perp})^*) = \det(\frac{1}{q}\Lambda) \hat{f}(\frac{1}{q}\Lambda)$. Set

$f(\boldsymbol{h}) = (\rho_s \cdot e_{j\boldsymbol{c}/q})(\boldsymbol{h}+\boldsymbol{h}_0)$. We have:

$$\hat{f} = \widehat{\rho_s \cdot e_{\boldsymbol{v}}} \cdot e_{\boldsymbol{h}_0} = s^m \rho_{1/s,\boldsymbol{v}} \cdot e_{\boldsymbol{h}_0} \ .$$

We proceed:

$$P = \frac{\det((\Lambda^{\perp})^*)s^m}{\rho_s(\Lambda^{\perp}_{\boldsymbol{p}})} \sum_{j\in\mathbb{Z}_q} \hat{r}_j \cdot (\rho_{1/s,j\boldsymbol{c}/q} \cdot e_{\boldsymbol{h}_0}) \left(\frac{1}{q}\Lambda\right)$$
$$P = \frac{\det((\Lambda^{\perp})^*)s^m}{\rho_s(\Lambda^{\perp}_{\boldsymbol{p}})} \sum_{j\in\mathbb{Z}_q} \hat{r}_j \cdot \sum_{\boldsymbol{y}\in\Lambda} (\rho_{q/s,j\boldsymbol{c}} \cdot e_{\boldsymbol{h}_0/q})\,(\boldsymbol{y})\,.$$

Assuming $s \geq \eta_\epsilon(\Lambda^\perp)$ for some negligible ϵ ensures that $\frac{\det((\Lambda^{\perp})^*)s^m}{\rho_s(\Lambda^{\perp}_{\boldsymbol{p}})} = 1 + O(\epsilon)$ by Lemma 2.3. We shall split the sum into three parts:

– $j = 0$, $\boldsymbol{y} = \boldsymbol{0}$, contributing exactly $\hat{r}_0$ (where $\hat{r}_0 = \frac{1}{q}\sum_{x\in\mathbb{Z}_q} r(x) \in [0,1]$),
– $j = 0$, $\boldsymbol{y} \neq \boldsymbol{0}$, contributing at most $|\hat{r}_0|\rho_{q/s}(\Lambda \setminus \{\boldsymbol{0}\})$ in absolute value,
– $j \neq 0$, contributing at most $|\hat{r}_j|\rho_{q/s}(\Lambda - j\boldsymbol{c})$ in absolute value for each j.

We can now bound P:

$$\left|\frac{P}{1-O(\epsilon)} - \hat{r}_0\right| \leq |\hat{r}_0|\rho_{q/s}(\Lambda \setminus \{\boldsymbol{0}\}) + \sum_{j\in\mathbb{Z}_q\setminus\{0\}} |\hat{r}_j|\rho_{q/s}(\Lambda - j\boldsymbol{c}) \ .$$

We now want to bound the right-hand side using Lemma 2.1, with $c = 1$ for simplicity. Fix $j \in \mathbb{Z}_q \setminus \{0\}$, and let $\alpha = q\sqrt{m}/s$. If $\alpha < d(j\boldsymbol{c}, \Lambda)$, then $(\Lambda - j\boldsymbol{c}) \setminus \alpha\mathcal{B} = (\Lambda - j\boldsymbol{c})$. Also, note that $\rho_{q/s}(\Lambda) = \rho_{1/s}(\frac{1}{q}\Lambda) = \rho_{1/s}((\Lambda^\perp)^*)$. So, as long as $s \geq \eta_\epsilon(\Lambda^\perp)$ for some negligible ϵ (which we already assumed earlier), it holds that $\rho_{q/s}(\Lambda) \leq 1 + \epsilon$ by definition of $\eta_\epsilon(\Lambda^\perp)$. Under those conditions, $\rho_{q/s}(\Lambda - j\boldsymbol{c}) = \rho_{q/s}((\Lambda - j\boldsymbol{c}) \setminus \alpha\mathcal{B}) \leq 2C^m\rho_{q/s}(\Lambda) \leq 2C^m(1+\epsilon)$ is negligible. Using Lemma 2.1, we deduce the following:

Theorem 3.1. *Fix $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$, $\boldsymbol{c} \in \mathbb{Z}_q^m$, and $\boldsymbol{p} \in \mathbb{Z}_q^n$, where m is polynomial in n. Fix a probabilistic rounding function $R : \mathbb{Z}_q \to \{0,1\}$ such that for all $x \in \mathbb{Z}_q$,*

$$\Pr[R(x) = 1] = r(x) = \sum_{j\in J} \hat{r}_j e_{j/q}(x) \ ,$$

where $J \subseteq \mathbb{Z}_q$ and $\hat{r}_j \in \mathbb{C}$. Let $s \geq \eta_\epsilon(\Lambda^\perp(\boldsymbol{A}))$ for some $\epsilon = \mathrm{negl}(n)$. Assume furthermore that

$$\forall j \in J \setminus \{0\}, \quad s \cdot d(j\boldsymbol{c}, \Lambda(\boldsymbol{A})) > q\sqrt{m} \ .$$

Denote $P(\boldsymbol{c}) = \Pr[R(\langle\boldsymbol{h}, \boldsymbol{c}\rangle) = 1 \mid \boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{p}^t]$, where the probability is taken over the randomness of R, and the distribution of $\boldsymbol{h} \leftarrow D^m_{\mathbb{Z},s}$, conditioned on $\boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{p}^t$. Then :

$$|P(\boldsymbol{c}) - \hat{r}_0| \leq (2 + O(\epsilon))\,|J|\,C^m + O(\epsilon) \quad \textit{where} \quad C = \sqrt{2\pi e} \cdot e^{-\pi} < 1 \ .$$

Setting up the rounding function. If one wishes to avoid having to attempt decryption of many multiples of the ciphertext $\boldsymbol{c}$, one should choose a probabilistic rounding function with a small number of harmonics.

In particular, the typical deterministic rounding function $R(x) = \lfloor 2x/q \rceil \bmod 2$ —the so-called square-signal— and has harmonic coefficients $\hat{r}_j$ decreasing as $\Theta(1/j)$ in absolute value (for odd $j \in \{\lceil -q/2 \rceil, \dots, \lfloor q/2 \rfloor\}$). With such a rounding function, one would still need to attempt trapdoor inversion for $q/2$ many multiples of $\boldsymbol{c}$, as it was already the case in [KV09].

On the contrary, one may easily avoid costly harmonics by setting the rounding function so that $2r(x) = 1 + \cos(2\pi x/q)$, which has Fourier coefficients $\hat{r}_0 = 1/2$, $\hat{r}_1 = \hat{r}_{-1} = 1/4$, and $\hat{r}_j = 0$ for any other j.[13] More precisely, we have the following corollary by remarking that when $\boldsymbol{c} \notin \mathscr{L}$ and $\alpha = q\sqrt{m}/s < B'$, we have $d(\boldsymbol{c}, \Lambda) \geq B'$ and $(\Lambda - \boldsymbol{c}) \setminus (\alpha\mathcal{B}) = (\Lambda - \boldsymbol{c})$.

Corollary 3.2. *Let $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$ with $m = \Theta(n \log q)$, and fix $\boldsymbol{p} \in \mathbb{Z}_q^n$. Let $B' = q/\Theta(\sqrt{m})$, and $\mathscr{L} = \{\boldsymbol{c} \in \mathbb{Z}_q^m \mid d(\boldsymbol{c}, \Lambda(\boldsymbol{A})) \leq B'\}$. Suppose that R satisfies:*

$$\Pr[R(x) = 1] = r(x) = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2\pi x}{q}\right) \tag{1}$$

and let $s \geq \eta_\epsilon(\Lambda^\perp(\boldsymbol{A}))$ for some $\epsilon = \mathrm{negl}(n)$. Suppose also that: $s > \frac{q\sqrt{m}}{B'}$.

Denote again $P(\boldsymbol{c}) = \Pr[R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle) = 1 \mid \boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{p}^t]$, where the probability is taken over the randomness of R, and the distribution of $\boldsymbol{h} \leftarrow D^m_{\mathbb{Z},s}$, conditioned on $\boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{p}^t$. Then, for all $\boldsymbol{c} \notin \mathscr{L}$:

$$|2P(\boldsymbol{c}) - 1| \leq 2\,(6 + O(\epsilon))\,C^m + O(\epsilon) \leq \mathrm{negl}(n) \ ,$$

where $C = \sqrt{2\pi e} \cdot e^{-\pi} < 1$.

3.3 Approximate Correctness

Let us check that the scheme above achieves approximate correctness, that is, for all $\boldsymbol{c} \in \widetilde{\mathscr{L}}$, $\mathsf{Hash}(\boldsymbol{h}, \boldsymbol{A}, \boldsymbol{c}) = \mathsf{ProjHash}(\boldsymbol{p}, \boldsymbol{A}, \boldsymbol{c}, (\boldsymbol{s}, \boldsymbol{e}))$ with probability substantially greater than 1/2. Using our rounding function R, this means that we want $R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle)$ and $R(\langle \boldsymbol{p}, \boldsymbol{s} \rangle)$ to output the same bit with some probability Q substantially greater than 1/2, where the two applications of R use independent coins.

Recall that $r(x)$ is the probability that the rounding function R outputs 1 on input x, and that for $\boldsymbol{c} \in \widetilde{\mathscr{L}}$, we can write $\langle \boldsymbol{h}, \boldsymbol{c} \rangle = \langle \boldsymbol{p}, \boldsymbol{s} \rangle + \langle \boldsymbol{h}, \boldsymbol{e} \rangle$, where $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$. We argue that as long as $\langle \boldsymbol{h}, \boldsymbol{e} \rangle$ is small with respect to q, then our scheme achieves approximate correctness:

[13] Of course, one could also obtain perfect universality by setting a constant rounding function $r(x) = 1/2$, and even avoid the first harmonic, but there is no way to reach correctness even with amplification in that case.

Lemma 3.3. *Fix $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ and $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \widetilde{\mathscr{L}}$, where m and q are polynomial in n, and where $\|\boldsymbol{e}\| \le B = 2t\sqrt{m}$. Let $s \ge \eta_\epsilon(\Lambda^\perp(\boldsymbol{A}))$ for some $\epsilon = \mathrm{negl}(n)$. Assume that R is the cosine rounding function (Eq. (1)). Let Q be the probability that $R(\langle \boldsymbol{A}^t\boldsymbol{h}, \boldsymbol{s}\rangle; \mathsf{coins}_1)$ and $R(\langle \boldsymbol{h}, \boldsymbol{c}\rangle; \mathsf{coins}_2)$ output the same bit, over the randomness of $\boldsymbol{h} \leftarrow D_{\mathbb{Z},s}^m$, and the randomness of the two independent coins coins_1 and coins_2 used by R. If $tsm = o(q)$, then $Q = 3/4 + o(1)$.*

Proof. As $s \ge \eta_\epsilon(\Lambda^\perp)$ for $\epsilon = \mathrm{negl}(n)$, the distribution of $\boldsymbol{h}^t\boldsymbol{A}$, when $\boldsymbol{h} \leftarrow D_{\mathbb{Z},s}^m$, is at negligible statistical distance from uniform.

Therefore, Q is negligibly close to $\Pr[R(x; \mathsf{coins}_1) = R(x + \langle \boldsymbol{h}, \boldsymbol{e}\rangle; \mathsf{coins}_2)]$ where the probability is taken over uniform $x \in \mathbb{Z}_q$, $\boldsymbol{h} \leftarrow D_{\mathbb{Z},s}^m$, and the randomness of the two independent coins coins_1 and coins_2 used by R.

Then:

$$\begin{aligned} Q &= \frac{1}{q}\sum_{x\in\mathbb{Z}_q} \left(r(x)r(x+\langle \boldsymbol{h}, \boldsymbol{e}\rangle) + (1-r(x))(1-r(x+\langle \boldsymbol{h}, \boldsymbol{e}\rangle))\right) + \mathrm{negl}(n) \\ &= \frac{1}{2} + \frac{1}{q}\sum_{x\in\mathbb{Z}_q} \frac{1}{2}\cos\left(2\pi\frac{x}{q}\right)\cos\left(2\pi\frac{x+\langle \boldsymbol{h}, \boldsymbol{e}\rangle}{q}\right) + \mathrm{negl}(n) \ . \end{aligned}$$

As $tsm = o(q)$, we have $\langle \boldsymbol{h}, \boldsymbol{e}\rangle = o(q)$ with overwhelming probability. As cos is a Lipschitz continuous function, we can approximate the sum by an integral:

$$Q = \frac{1}{2} + \frac{1}{2}\int_0^1 \cos^2(2\pi x)dx + o(1) = \frac{3}{4} + o(1) \ .$$

This concludes the proof. □

3.4 Wrap-up

Consider the bit-PHF described in Section 3.1 instantiating R with the cosine rounding function (Eq. (1)), together with the encryption scheme of Section 2.2. Let us now show that all the parameters can be instantiated to satisfy security and correctness of the encryption scheme, simultaneously with statistical universality and approximate correctness of the bit-PHF.

IND-CCA2. To base the security of the scheme described in Section 2.2 on $\mathrm{LWE}_{\chi,q}$ for $\chi = D_{\mathbb{Z},\sigma}$ and $\sigma = 2\sqrt{n}$,[14] we apply Theorem 2.9 with $m = \Theta(n \log q)$ and $t = \sqrt{mn}\cdot\omega(\sqrt{\log n})$.

Decryption Correctness. For the encryption scheme to be correct, we want $B < B'$, recalling that $B := 2t\sqrt{m}$ and $B' := q/\Theta(\sqrt{m})$.

Universality. In Corollary 3.2, we used the hypothesis $s \ge \eta_\epsilon(\Lambda^\perp(\boldsymbol{A}_u))$ for some negligible ϵ. Assuming $s \ge \Theta(\sqrt{n})$, one can apply Lemma 2.11, to ensure the

[14] This is the smallest parameter σ for which $\mathrm{LWE}_{\chi,q}$ is known reduce to a worst-case problem. One may of course choose to use a different width for the LWE error, and derive different appropriate parameters.

above hypothesis for $\epsilon = 2^{-n}$ simultaneously for all $u \in \mathcal{U}$ except with probability 2^{-n} over the randomness of $\mathsf{TrapGen}$.

Still in Corollary 3.2, we also needed $s > q\sqrt{m}/B'$, where $B' = q/\Theta(\sqrt{m})$. This holds for $s = \Theta(m)$.

Approximate correctness. For Lemma 3.3, we assumed that $tsm = o(q)$. Equivalently, it is sufficient that $sm^{3/2}n^{1/2}\omega(\sqrt{\log n}) = o(q)$.

Summary. Therefore, all the desired conditions can be satisfied with $q = \tilde{\Theta}(n^3)$, $m = \tilde{\Theta}(n)$, $s = \tilde{\Theta}(n)$, and $t = \tilde{\Theta}(n)$. We have proved the following:

Theorem 3.4. *Set parameters $q = \tilde{\Theta}(n^3), m = \tilde{\Theta}(n), s = \tilde{\Theta}(n), t = \tilde{\Theta}(n)$. Define a probabilistic rounding function $R : \mathbb{Z}_q \to \{0,1\}$ such that $\Pr[R(x) = 1] = 1/2 + \cos(2\pi x/q)/2$. Then, i) the encryption scheme of Section 2.2 is correct and tag-IND-CCA2 under the hardness of LWE$_{\chi,q}$ for $\chi = D_{\mathbb{Z},2\sqrt{n}}$; and ii) the bit-PHF described in Section 3.1 achieves statistical universality and $(1/4 - o(1))$-correctness.*

4 Word-Independent SPHF for IND-CPA Ciphertexts

4.1 Overview

In the previous section, we built a bit-PHF with negl(n)-universality but approximate correctness. Even though correctness can be amplified (as described in Appendix B.2), the transformation inherently makes the new projection key depend on the word we want to hash, even if that was not the case for the initial bit-PHF.

We now build a bit-PHF with statistical correctness and K-universality for some universal constant $K < 1$ (but using a super-polynomial LWE modulus q). The main benefit of such a construction is that amplifying universality can be done regardless of the word we want to hash, that is, the projection key will not depend on the word (see Lemma B.4). When the tag u of the ciphertext $\boldsymbol{c}$ is known in advance or is constant (in which case, the encryption scheme is only IND-CPA instead of IND-CCA2), we therefore get a word-independent bit-PHF which can be transformed into a word-independent SPHF. This is the first word-independent SPHF for any lattice-based language.

We use the same natural approach as described in Section 3.1. The only differences with the construction in the previous section are the probabilistic rounding function we use, and the parameters necessary to argue correctness and universality. Recall that in the last section, we used a rounding function with only low order harmonics to get negl(n)-universality.

The starting point is the observation that, for the naive square rounding introduced in the previous section, the correctness is statistical, but clearly not negl(n)-universal, depending on which word $\boldsymbol{c}$ is hashed (as seen in the two case studies in the technical overview in the introduction, where $j \cdot \boldsymbol{c}$ is close to Λ for some $j \in \mathbb{Z}_q^*$). However, the distribution of $R(\langle \boldsymbol{h}, \boldsymbol{c}\rangle)$ conditioned on $\boldsymbol{h}^t\boldsymbol{A}$ might still have enough entropy to give us K-universality, for some constant $K < 1$. In other words, we can hope that $|2 \cdot \Pr[R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle) = 1 \mid \boldsymbol{p}] - 1| \leq K$ for all $\boldsymbol{c} \in \mathbb{Z}_q^m$.

Let $R^\sharp$ be a rounding function defined by: $R^\sharp(x) = 1 + \lfloor 2x/q \rceil \bmod 2$, that is:

$$\forall x \in [-q/2, q/2], \quad R^\sharp(x) = \begin{cases} 1 & \text{if } |x| \in [-q/4, q/4) \\ 0 & \text{otherwise.} \end{cases} ,$$

Using this rounding function gives good correctness: when $s \geq \eta_\epsilon(\Lambda^\perp)$, $\langle \boldsymbol{h}, \boldsymbol{c} \rangle$ is statistically close to uniform in $[-q/2, q/2]$, and therefore $R^\sharp(\langle \boldsymbol{h}, \boldsymbol{c} \rangle)$ is a uniform bit up to some statistical distance $O(\epsilon + 1/q)$ (due to the fact that q is odd). So for super-polynomial q, we get *statistical correctness* using $R^\sharp$ as rounding function, as long as $\langle \boldsymbol{h}, \boldsymbol{e} \rangle$ is sufficiently small with respect to q.

For *universality*, we express the probability distribution defined by $R^\sharp$, seen as a q-periodic function over $\mathbb{R}$, as a Fourier series:

$$\forall x \in [-q/2, q/2], \quad r^\sharp(x) \coloneqq \Pr[R^\sharp(x) = 1] = \sum_{j \in \mathbb{Z}} \hat{r}_j^\sharp \cdot e_{j/q}(x) ,$$

where $\hat{r}_j^\sharp$ are the Fourier coefficients of the q-periodic function $r^\sharp \colon \mathbb{R} \to \mathbb{R}$.

However, one can show that $|\hat{r}_j^\sharp| = \Theta(1/j)$ (for odd integers j). Therefore, it is not clear how to show universality with a similar analysis as in Section 3.2: the total contribution of harmonics j such that $j \cdot \boldsymbol{c}$ is close to Λ could potentially be arbitrarily large!

To solve this issue, we consider a new rounding function R, which has the same probability distribution as $R^\sharp$ but on a negligible fraction of integer points (so that statistical correctness is preserved), and such that its Fourier coefficients of high enough order have small enough amplitude.

Then, we use the observation that the set of integers j such that $j \cdot \boldsymbol{c}$ is in Λ is an ideal of $\mathbb{Z}$, which is proper if $\boldsymbol{c}$ itself is not in Λ. More generally, the set of *small* integers $j \in \mathbb{Z}$ such that $j \cdot \boldsymbol{c}$ is *close* to Λ is contained in an ideal of $\mathbb{Z}$; furthermore, if $\boldsymbol{c}$ is far from Λ, then the smallest such ideal is a proper ideal of $\mathbb{Z}$. This will allow us to discard all harmonics whose order is not in this ideal. As we will show, the remaining harmonics necessarily have destructive interferences, which allows us to establish K-universality for some constant $K < 1$.

The roadmap follows. First, in Section 4.2, we smooth the discontinuities of the probability distribution of the square rounding function $r^\sharp$ so that the Fourier coefficients of high order have small magnitude, but such that we keep statistical correctness. Then to prove universality, in Section 4.3, we show that for $\boldsymbol{c}$ far from Λ, the set of small $j \in \mathbb{Z}$ such that $j \cdot \boldsymbol{c}$ is close to Λ is contained in a proper ideal of $\mathbb{Z}$. Finally, in Section 4.4 we show that the distribution of $R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle)$ conditioned on $\boldsymbol{h}^t \boldsymbol{A}$ has some bounded min entropy.

4.2 Smoothing the Discontinuities: a New Rounding Function

In the following, unless specified otherwise, we will see $\mathbb{Z}_q$ as embedded in $\{\lceil -q/2 \rceil, \ldots, \lfloor q/2 \rfloor\}$, and the canonical period we use for q-periodic functions will be $[-q/2, q/2]$. Recall that $r^\sharp$ satisfies:

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

In particular, $r^\sharp$ has two discontinuities on $q/4$ and on $-q/4$. To smooth those discontinuities, we consider the convolution product of the square signal $r^\sharp$ with a rectangular signal of appropriate width T such that $T/q = \mathrm{negl}(n)$. More precisely, consider the q-periodic function $r^\flat$ defined on $[-q/2, q/2]$ by:

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

We define a new rounding function R such that for all $x \in \mathbb{R}$ (see Fig. 1):

$$\Pr[R(x) = 1] := r(x) := (r^\sharp \odot r^\flat)(x) := \int_{-q/2}^{q/2} r^\sharp(u) \cdot r^\flat(x-u)\, du \enspace ,$$

where, in this context, $\odot$ corresponds to the convolution of q-periodic functions.

Intuitively, this corresponds to replace the discontinuities on $r^\sharp(\pm q/4)$ by a linear slope ranging from $\pm q/4 - T$ to $\pm q/4 + T$ (see Fig. 1 on page 6). Therefore, over $[-q/2, q/2]$, the functions r and $r^\sharp$ only differ on at most $4\lceil T \rceil$ integer points (the points on the slope). Recall that if $s \ge \eta_\epsilon(\Lambda^\perp)$ for some negligible ϵ, then $\langle \boldsymbol{h}, \boldsymbol{c} \rangle$ is statistically close to uniform in $\{\lceil -q/2 \rceil, \ldots, \lfloor q/2 \rfloor\}$. Therefore, if $\langle \boldsymbol{h}, \boldsymbol{e} \rangle / q$ and T/q are negligible, then:

$$\Pr[R(\langle \boldsymbol{h}, \boldsymbol{c} \rangle) \neq R(\langle \boldsymbol{p}, \boldsymbol{s} \rangle)] \le \mathrm{negl}(n) \enspace ,$$

and we get statistical correctness using such a rounding function.

Lemma 4.1 (Correctness). *Suppose that $s \ge \eta_\epsilon(\Lambda^\perp)$ for some $\epsilon = \mathrm{negl}(n)$, $tsm/q = \mathrm{negl}(n)$, and $T/q = \mathrm{negl}(n)$. Assume that R satisfies: $\Pr[R(x) = 1] = r(x) = (r^\sharp \odot r^\flat)(x)$. Then the approximate bit-PHF defined in Section 3.1 achieves statistical correctness.*

Furthermore, r is q-periodic, and can therefore be expressed as a Fourier series:

$$\forall x \in [-q/2, q/2], \quad r(x) = \sum_{j \in \mathbb{Z}} \hat{r}_j e_{j/q}(x) \enspace ,$$

with Fourier coefficients $\hat{r}_j$. As $r = r^\sharp \odot r^\flat$, we have $\hat{r}_j = q \cdot \hat{r}^\sharp_j \cdot \hat{r}^\flat_j$ for $j \in \mathbb{Z}$, where $\hat{r}^\sharp_j$ and $\hat{r}^\flat_j$ are the Fourier coefficients of the q-periodic functions $r^\sharp$ and $r^\flat$ respectively. Thus, $\hat{r}_0 = 1/2$, and for $j \in \mathbb{Z} \setminus \{0\}$, the jth harmonic of r is:

$$\hat{r}_j = \frac{q}{2\pi^2 T j^2} \cdot \sin(\pi j/2) \cdot \sin(2\pi T j/q) \le \frac{q}{19 T j^2} \enspace . \tag{2}$$

4.3 Inclusion of Contributing Harmonics in a Proper Ideal

In the following, we focus on showing that even though we do not have negl(n)-universality using this new rounding function, we still have some K-universality for some constant $K < 1$ (that we can amplify).

We start by a simple useful lemma:

Lemma 4.2. *Let $N = kq/T$ for some k. Then $\sum_{j\in\mathbb{Z},\ |j|>N} |\hat{r}_j| \leq 1/k$.*

Proof. It follows from Eq. (2) and the fact that for all $N > 2$: $\sum_{k=N}^{+\infty} \frac{1}{k^2} \leq \sum_{k=N}^{+\infty} \left(\frac{1}{k-1} - \frac{1}{k}\right) = \frac{1}{N-1}$. □

Suppose now that $d(\boldsymbol{c}, \Lambda) \geq B'$. Consider the set of $j \in \mathbb{Z}$ such that $d(j\cdot\boldsymbol{c}, \Lambda) \leq \delta$ for some appropriately chosen δ. Let $P = P(\boldsymbol{c}) = \Pr[R(\langle \boldsymbol{h}, \boldsymbol{c}\rangle) = 1 \mid \boldsymbol{h}^t \boldsymbol{A} = \boldsymbol{p}^t]$, for our new rounding function R. For any $\boldsymbol{h_0} \in \Lambda^{\perp}_{\boldsymbol{p}}$, we can show similarly to Section 3.2, that:

$$P = \frac{\det((\Lambda^{\perp})^*)s^m}{\rho_s(\Lambda^{\perp}_{\boldsymbol{p}})} \sum_{j\in\mathbb{Z}} \hat{r}_j \sum_{y\in\Lambda} (\rho_{q/s,jc} \cdot e_{\boldsymbol{h}_0/q})(\boldsymbol{y}) \ , \tag{3}$$

where $\frac{\det((\Lambda^{\perp})^*)s^m}{\rho_s(\Lambda^{\perp}_{\boldsymbol{p}})} = (1 + O(\epsilon))$ as long as $s \geq \eta_\epsilon(\Lambda^{\perp})$. Note that $\sum_{|j|\geq N} |\hat{r}_j|$ can be made arbitrarily small for appropriate N, by Lemma 4.2. Thus only the terms of the sum corresponding to $|j| \leq N$ will have a substantial contribution to the sum above (recall that $\rho_{q/s}(\Lambda - j\boldsymbol{c}) \leq 1 + \epsilon$ for all $\boldsymbol{c}$, for appropriate parameters). Therefore we only consider those small j such that $|j| < N$ for some appropriately chosen N (with respect to q). Furthermore, for large enough δ, the terms corresponding to indices j such that $d(j \cdot \boldsymbol{c}, \Lambda) > \delta$ also have a negligible contribution to the sum by Lemma 2.1. For appropriate parameters N and δ to be instantiated later, let:

$$J = \{j \in \mathbb{Z} \mid |j| < N \wedge d(j \cdot \boldsymbol{c}, \Lambda) \leq \delta\} \ . \tag{4}$$

As a subset of $\mathbb{Z}$, J is contained in the ideal $j_0\mathbb{Z}$ of $\mathbb{Z}$, where $j_0 = \gcd(J)$. Let us show that it is a proper ideal of $\mathbb{Z}$, i.e., $j_0 \neq 1$. To do so, we rely on the existence of small Bézout coefficients.

Lemma 4.3 (Corollary of [MH94, Theorem 9]). *Let $a_1, \ldots, a_k \in \mathbb{Z}$, and let $g = \gcd(a_1, \ldots, a_k)$. Then there exists $u_1, \ldots, u_k \in \mathbb{Z}$ such that the following conditions hold:*

$$\sum_{i=1}^{k} u_i a_i = g \ , \qquad \sum_{i=1}^{k} |u_i| \leq \frac{k}{2} \max |a_i| \ .$$

We can now prove that J is a proper ideal of $\mathbb{Z}$:

Lemma 4.4. *Suppose that $\delta N^2 < B'$. Then, for $\boldsymbol{c} \in \mathbb{Z}_q^m$ such that $d(\boldsymbol{c}, \Lambda) > B'$, the set $J = \{j \in \mathbb{Z} \mid |j| < N \wedge d(j \cdot \boldsymbol{c}, \Lambda) \leq \delta\}$ is contained in a proper ideal of $\mathbb{Z}$.*

Proof. Let $j_0 = \gcd(J)$. By definition, $J \subseteq j_0\mathbb{Z}$. Suppose by contradiction that $j_0 = 1$. By Lemma 4.3, there exists a set of integers $\{u_j, j \in J\}$ such that $\sum_{j\in J} u_j \cdot j = 1$ and then $\sum_{j\in J} u_j \cdot (j \cdot \boldsymbol{c}) = \boldsymbol{c}$. But by definition of J, $d(j \cdot \boldsymbol{c}, \Lambda) \leq \delta$ for all $j \in J$, and therefore:

$$d(\boldsymbol{c}, \Lambda) \leq \delta \cdot \sum_{j\in J} |u_j| \leq \frac{\delta \cdot |J|}{2} \max_{j\in J} |j| \leq \delta N^2 < B' \ ,$$

which is absurd as we assumed $d(\boldsymbol{c}, \Lambda) > B'$. □

4.4 Imperfect Universality from Destructive Interferences

We now want to quantify how biased $R(\langle \boldsymbol{h}, \boldsymbol{c}\rangle)$ conditioned on $\boldsymbol{h}^t\boldsymbol{A}$ can be when $\boldsymbol{c}$ is far from Λ. We start from Eq. (3):

$$P = \frac{\det((\Lambda^\perp)^*)s^m}{\rho_s(\Lambda^\perp_{\boldsymbol{p}})} \sum_{j\in\mathbb{Z}} \hat{r}_j \sum_{y\in\Lambda} (\rho_{q/s,jc} \cdot e_{\boldsymbol{h}_0/q})(\boldsymbol{y}) \ ,$$

where $\frac{\det((\Lambda^\perp)^*)s^m}{\rho_s(\Lambda^\perp_{\boldsymbol{p}})} = 1 + O(\epsilon)$ as long as $s \geq \eta_\epsilon(\Lambda^\perp)$.

We split the sum into three parts $P = P_1 + P_2 + P_3$:

P_1. $|j| > N \wedge j \notin j_0\mathbb{Z}$: those indices have a negligible contribution to the sum by Lemma 4.2.

P_2. $|j| \leq N \wedge j \notin j_0\mathbb{Z}$: those indices contribute negligibly since $\rho_{q/s}(\Lambda - j\boldsymbol{c})$ is small as $j\boldsymbol{c}$ is far from Λ (by definition of δ and $J \subset j_0\mathbb{Z}$).

P_3. $j \in j_0\mathbb{Z}$: the contributing terms. Unlike the previous ones we won't use absolute bounds for each term, and must consider destructive interferences.

It remains to study P_3, for which a similar computation as in Section 3.2 gives:

$$\begin{aligned} P_3 &= \frac{\det((\Lambda^\perp)^*)s^m}{\rho_s(\Lambda^\perp_{\boldsymbol{p}})} \sum_{j\in j_0\mathbb{Z}} \hat{r}_j \sum_{y\in\Lambda} (\rho_{q/s,jc} \cdot e_{\boldsymbol{h}_0/q})(\boldsymbol{y}) \\ &= \sum_{\boldsymbol{h}\in\Lambda^\perp_{\boldsymbol{p}}} \frac{\rho_s(\boldsymbol{h})}{\rho_s(\Lambda^\perp_{\boldsymbol{p}})} \sum_{j\in j_0\mathbb{Z}} \hat{r}_j e_{j/q}(\langle \boldsymbol{h}, \boldsymbol{c}\rangle) \ . \end{aligned}$$

If we were to have $j_0 = 1$ (i.e. $j_0\mathbb{Z} = \mathbb{Z}$), we could compute the inner sum simply by inverse Fourier transform, evaluating r at $x = \langle \boldsymbol{h}, \boldsymbol{c}\rangle$. Instead, we note that selecting only the harmonics in $j_0\mathbb{Z}$, corresponds in the temporal domain to averaging the function r over all its temporal shifts by multiples of q/j_0. More formally, recall the identity:

$$\sum_{k=0}^{j_0-1} e_{j/j_0}(k) = \begin{cases} j_0 & \text{if } j \in j_0\mathbb{Z} \\ 0 & \text{otherwise.} \end{cases}$$

We may now rewrite:

$$\sum_{j \in j_0\mathbb{Z}} \widehat{r}_j e_{j/q}(x) = \frac{1}{j_0} \sum_{j \in \mathbb{Z}} \widehat{r}_j e_{j/q}(x) \sum_{k=0}^{j_0-1} e_{j/j_0}(k) = \frac{1}{j_0} \sum_{k=0}^{j_0-1} r(x + k\frac{q}{j_0}) \ ,$$

Note that $\frac{1}{j_0}\sum_{k=0}^{j_0-1} r^\sharp(x + k\frac{q}{j_0})$ is not too far away from $1/2$: if j_0 is even, this is exactly $1/2$ (for all x), and if $j_0 = 2k+1$, this is either k/j_0 or $(k+1)/j_0$ (depending on x), which is at distance $1/(2j_0) \leq 1/6$ from $1/2$ (recall that $j_0 > 1$ by Lemma 4.4). Furthermore, we have:

$$\forall x \in [-q/2, q/2],\ r(x) = \frac{1}{2T}\int_{-T}^{T} r^\sharp(x+u)du \ ,$$

which gives, for all $x \in [-q/2, q/2]$:

$$\left|\frac{1}{j_0}\sum_{k=0}^{j_0-1} r(x + k\frac{q}{j_0}) - \frac{1}{2}\right| \leq \frac{1}{2T}\int_{-T}^{T}\left|\frac{1}{j_0}\sum_{k=0}^{j_0-1} r^\sharp(x + u + k\frac{q}{j_0}) - \frac{1}{2}\right| du \leq 1/6 \ .$$

Therefore, P_3 is also not too far from $1/2$ as a convex combination of values not too far from $1/2$. More precisely we have $|P_3 - 1/2| \leq 1/6$.

Putting everything together, we can quantify the distance from P to $1/2$:

Theorem 4.5 (Universality). *Let $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ with $m = \Theta(n \log q)$, and fix $\boldsymbol{p} \in \mathbb{Z}_q^n$. Let $B' = q/\Theta(\sqrt{m})$, and $\mathscr{L} = \{\boldsymbol{c} \in \mathbb{Z}_q^m \mid d(\boldsymbol{c}, \Lambda(\boldsymbol{A})) \leq B'\}$. Let R be as defined in Section 4.2 and let $s \geq \eta_\epsilon(\Lambda^\perp(\boldsymbol{A}))$ for some $\epsilon = \mathrm{negl}(n)$. Suppose also that parameters T, N, δ, and k satisfy $\delta > \frac{q\sqrt{m}}{s}$, $N = \frac{kq}{T}$, and $\delta N^2 < B'$.*

Denote again $P(\boldsymbol{c}) = \Pr[R(\langle \boldsymbol{h}, \boldsymbol{c}\rangle) = 1 \mid \boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{p}^t]$, where the probability is taken over the randomness of R, and the distribution of $\boldsymbol{h} \leftarrow D_{\mathbb{Z},s}^m$, conditioned on $\boldsymbol{h}^t\boldsymbol{A} = \boldsymbol{p}^t$. Then, for all $\boldsymbol{c} \notin \mathscr{L}$:

$$|P(\boldsymbol{c}) - 1/2| \leq \frac{1}{6} + (1 + O(\epsilon))\left(\frac{1}{k} + 4NC^m\right) \ ,$$

where $C = \sqrt{2\pi e}\cdot e^{-\pi} < 1$.

Remark 4.6. Informally, this theorem states that the second case study of the technical overview of the introduction is essentially the worst case.

Proof. Writing $P = P_1 + P_2 + P_3$ as above, we showed that $|P_3 - 1/2| \leq 1/6$. Moreover, as $s \geq \eta_\epsilon(\Lambda^\perp(\boldsymbol{A}))$, we have:

$$\frac{\det((\Lambda^\perp)^*)s^m}{\rho_s(\Lambda^\perp_{\boldsymbol{p}})} = 1 + O(\epsilon) \ ,$$

and, for any $j \in \mathbb{Z}$ and $\boldsymbol{c}$, we also have:

$$\left|\sum_{y \in \Lambda}(\rho_{q/s, j\boldsymbol{c}} \cdot e_{\boldsymbol{h}_0/q})(\boldsymbol{y})\right| \leq \rho_{q/s}(\Lambda - j\boldsymbol{c}) \leq 1 + \epsilon \ .$$

Therefore, by Lemma 4.2, and as $\epsilon = \mathrm{negl}(n)$, we have:

$$|P_1| \leq (1+O(\epsilon))(1+\epsilon) \sum_{|j|>N} |\hat{r}_j| \leq \frac{1+O(\epsilon)}{k} \ .$$

Furthermore, as $\delta > \frac{q\sqrt{m}}{s}$, and $|\hat{r}_j| \leq 1$ for all j, Lemma 2.1 gives us that $|P_2| \leq 4NC^m(1+O(\epsilon))$, which concludes the proof. □

4.5 Wrap-up

Let us now show that all the parameters can be instantiated to get approximate smoothness and correctness for the SPHF, using a rounding function R defined by $\Pr[R(x)=1] = r^\sharp \odot r^\flat(x)$.

IND-CPA. To apply Theorem 2.9 with Remark 2.10, we can use the fact that $m = \Theta(n \log q)$ and $t = \sqrt{mn} \cdot \omega(\sqrt{\log n})$.

Decryption Correctness. For the encryption scheme to be correct, we want $B < B'$, with $B = 2t\sqrt{m}$ and $B' = q/\Theta(\sqrt{m})$.

Correctness. For correctness of the bit-PHF, we need a super-polynomial modulus q, and require T/q to be negligible. Furthermore, we need tsm/q to be negligible, so that $\langle \boldsymbol{h}, \boldsymbol{e} \rangle$ can only take a negligible fraction of values in $\mathbb{Z}_q$. Also, we need $s \geq \eta_\epsilon(\Lambda^\perp(\boldsymbol{A}_u))$, which is satisfied with high probability by Lemma 2.11 for $\epsilon = 2^{-n}$ as long as $s \geq \Theta(\sqrt{n})$.

Bounding the amplitude of high frequencies. The parameter N which upper bounds the elements of J must be taken so that $\sum_{|j|\geq N} |\hat{r}_j|$ is small. By Lemma 4.2, by taking $N = kq/T$, this sum is $\leq 1/k$.

Threshold distance to Λ defining J. The parameter δ, which denotes how close $j \cdot \boldsymbol{c}$ is close to Λ for $j \in J$ (Eq. (4)) has to be chosen so that $N \cdot \rho_{q/s}(\Lambda - \boldsymbol{v})$ must be small whenever $d(\boldsymbol{v}, \Lambda) \geq \delta$. As in the analysis for the cosine rounding function, setting $\delta = q\sqrt{m}/s$ implies that $\rho_{q/s}(\Lambda - \boldsymbol{v}) \leq 2C^m(1+O(\epsilon))$ by Lemma 2.1.

Showing that $j_0 \neq 1$. We also required $\delta N^2 < B'$ to conclude that J was included in a proper ideal of $\mathbb{Z}$. As we have $\delta N^2 = \Theta\left(\frac{q^3 k \sqrt{m}}{sT^2}\right)$, this holds as long as $s \geq \Omega(\frac{mk^2q^2}{T^2})$.

Putting everything together, we get the following theorem:

Theorem 4.7. *Suppose $q = O(2^n)$ is superpolynomial in n, $m = \Theta(n \log q)$. Set parameters: i) T such that T/q and q/T^2 are both negligible in n (using $T = q^{2/3}$ for instance), ii) $k = \Theta(n)$, and iii) $s \geq \Theta(\sqrt{n})$ such that $s/q = \mathrm{negl}(n)$ and $s = \Omega(\frac{mk^2q^2}{T^2})$, which exists by construction of T. Define a probabilistic rounding function $R : \mathbb{Z}_q \to \{0,1\}$ such that $\Pr[R(x)=1] = r^\sharp \odot r^\flat(x)$. Then the bit-PHF described in Section 3.1 achieves $(1/3 + o(1))$-universality and statistical correctness.*

Proof. The theorem follows from the discussion above and Theorem 4.5 using: *i)* $N = kq/T$ (in which case NC^m is negligible in n), and *ii)* $\delta = \frac{q\sqrt{m}}{s}$. □

5 Applications

In this section, we present several applications of our new construction. It underlines the importance of revisiting this primitive.

5.1 Password-Authenticated Key Exchange

3-round PAKE. Gennaro and Lindell proposed in [GL06] a generic framework for building 3-round PAKE protocols based on an IND-CCA2 encryption scheme and an associated SPHF. Later in [KV09], Katz and Vaikuntanathan refined it to be compatible with approximate SPHF over a CCA2-secure encryption scheme.

We can instantiate the construction in [KV09] using the encryption scheme à la Micciancio-Peikert in Section 2.2 together with an approximate SPHF generically derived (via the transformation in Appendix B.2) from the approximate bit-PHF constructed in Section 3. This allows us to achieve a PAKE protocol in three flows, with a polynomial modulus.

Moving to a 2-round PAKE. An interesting optimization in cryptography is to reduce the number of rounds, so that each user only has to speak once. Is it possible to achieve a PAKE, where each user sends simply one flow?

In [ABP15b], the authors revisited the Groce-Katz framework [GK10]. Their construction (called GK-PAKE) uses a pseudo-random generator, an IND-CPA encryption scheme, with a simple regular SPHF on one hand, and an IND-PCA (Indistinguishable against Plaintext-Checkable Attacks) encryption on the other.

Every IND-CCA2 encryption being also IND-PCA, we can trivially meet the requirements and achieve the expected 2-rounds efficiency, using our SPHF from Section 3.[15] Contrary to the construction of Zhang and Yu [ZY17], we do not need a simulation-sound non-interactive proof (SS-NIZK), which we do not know how to construct from lattice assumptions in the standard model.

Achieving a 1-round PAKE. Actually, if we allow ourselves to use SS-NIZK, we can construct a 1-round PAKE by combining our word-independent SPHF with the ideas in [KV11], which solves an open problem in [ZY17]. Concretely, we use the first instantiation of [KV11], except that the ElGamal encryption scheme and its associated SPHF are replaced by our IND-CPA LWE-based encryption scheme à la Micciancio-Peikert and the word-independent SPHF is the one from Section 4. The SS-NIZK can be a simple variant of the one in [ZY17]. Details are provided in Appendix C.1.

[15] In this application, as in our 3-round PAKE from [KV09], the gap between correctness and smoothness is not an issue: the proof of the resulting 2-round PAKE works exactly as in [ABP15b].

5.2 Honest-Verifier Zero-Knowledge

Following the methodology from [BCPW15], using our SPHF in Section 3, we can construct honest-verifier zero-knowledge proofs for any NP language of the form $\ddot{\mathscr{L}} = \{\ddot{\chi} \mid \exists \ddot{w}, \ddot{\mathscr{R}}(\ddot{\chi}, \ddot{w})\}$ where $\ddot{\mathscr{R}}$ is a polynomial-size circuit. At a very high level, the prover simply encrypts each wire of the circuit using an IND-CPA encryption scheme[16] and then shows the correct evaluation at each gate, using SPHFs.

For the sake of simplicity, we suppose that all gates of the circuit $\ddot{\mathscr{R}}$ are NAND gates. We just need to construct an SPHF for the languages $\widetilde{\mathscr{L}} \subseteq \mathscr{L}$ of ciphertexts C_1, C_2, C_3 encrypting values (b_1, b_2, b_3) so that $b_3 = \mathrm{NAND}(b_1, b_2)$, such that $\widetilde{\mathscr{L}}$ is the set of encryptions of b_i that fits the NAND gate evaluation, while $\mathscr{L}$ is the set of ciphertexts whose decryptions fit the gate evaluation. We can do that by combining our SPHFs using the classical techniques described in [ACP09]. Details are provided in Appendix C.2.

5.3 Witness Encryption

Witness encryption [GGSW13] allows to encrypt a message, with respect to a particular word χ and a language $\mathscr{L}$, instead of using a classical public key. If the word is in the language, then a user knowing a witness for the word can decrypt the ciphertext, otherwise the ciphertext hides the message.

An SPHF can be used to construct such a primitive as follows: To encrypt a message M with respect to a word χ and a language $\mathscr{L}$, use an SPHF for $\mathscr{L}$ to generate a hashing key hk, a projection key hp, and a hash value H, and output the *ciphertext* $C = (\mathsf{hp}, \mathsf{H} \oplus M)$. To decrypt such a ciphertext, simply use the witness w associated with the word χ together with the projection key hp to compute the projected hash value and recover M. Details are available in Appendix C.3.

Acknowledgments

We would like to sincerely thank Zvika Brakerski for many useful and interesting discussions.

References

ABP15a. M. Abdalla, F. Benhamouda, and D. Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In *EUROCRYPT 2015, Part II*, *LNCS* 9057, pages 69–100. Springer, Heidelberg, April 2015.

ABP15b. M. Abdalla, F. Benhamouda, and D. Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In *PKC 2015*, *LNCS* 9020, pages 332–352. Springer, Heidelberg, March / April 2015.

[16] We actually will use our IND-CCA2 encryption scheme à la Micciancio-Peikert.

ACP09. M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *CRYPTO 2009*, *LNCS* 5677, pages 671–689. Springer, Heidelberg, August 2009.

Ban93. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

BBC$^+$13a. F. Ben Hamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In *PKC 2013*, *LNCS* 7778, pages 272–291. Springer, Heidelberg, February / March 2013.

BBC$^+$13b. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 449–475. Springer, Heidelberg, August 2013.

BCPW15. F. Benhamouda, G. Couteau, D. Pointcheval, and H. Wee. Implicit zero-knowledge arguments and applications to the malicious setting. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 107–129. Springer, Heidelberg, August 2015.

BH15. M. Bellare and V. T. Hoang. Adaptive witness encryption and asymmetric password-based cryptography. In *PKC 2015*, *LNCS* 9020, pages 308–331. Springer, Heidelberg, March / April 2015.

Boy13. X. Boyen. Attribute-based functional encryption on lattices. In *TCC 2013*, *LNCS* 7785, pages 122–142. Springer, Heidelberg, March 2013.

CHKP10. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, *LNCS* 6110, pages 523–552. Springer, Heidelberg, May 2010.

CS98. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*, *LNCS* 1462, pages 13–25. Springer, Heidelberg, August 1998.

CS02. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, *LNCS* 2332, pages 45–64. Springer, Heidelberg, April / May 2002.

DDN03. D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM review*, 45(4):727–784, 2003.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, *LNCS* 263, pages 186–194. Springer, Heidelberg, August 1987.

GGSW13. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

GK10. A. Groce and J. Katz. A new framework for efficient password-based authenticated key exchange. In *ACM CCS 10*, pages 516–525. ACM Press, October 2010.

GL06. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. *ACM Transactions on Information and System Security*, 9(2):181–234, 2006.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

GVW13. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

GVW15. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 503–523. Springer, Heidelberg, August 2015.

HK12. S. Halevi and Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012.

JG04. S. Jiang and G. Gong. Password based key exchange with mutual authentication. In *SAC 2004*, *LNCS* 3357, pages 267–279. Springer, Heidelberg, August 2004.

JR12. C. S. Jutla and A. Roy. Relatively-sound NIZKs and password-based key-exchange. In *PKC 2012*, *LNCS* 7293, pages 485–503. Springer, Heidelberg, May 2012.

Kal05. Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In *EUROCRYPT 2005*, *LNCS* 3494, pages 78–95. Springer, Heidelberg, May 2005.

KV09. J. Katz and V. Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In *ASIACRYPT 2009*, *LNCS* 5912, pages 636–652. Springer, Heidelberg, December 2009.

KV11. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC 2011*, *LNCS* 6597, pages 293–310. Springer, Heidelberg, March 2011.

Lyu08. V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC 2008*, *LNCS* 4939, pages 162–179. Springer, Heidelberg, March 2008.

Lyu09. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, *LNCS* 5912, pages 598–616. Springer, Heidelberg, December 2009.

MH94. B. S. Majewski and G. Havas. The complexity of greatest common divisor computations. In *International Algorithmic Number Theory Symposium*, pages 184–193. Springer, 1994.

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, *LNCS* 7237, pages 700–718. Springer, Heidelberg, April 2012.

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

NY90. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

Pei10. C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO 2010*, *LNCS* 6223, pages 80–97. Springer, Heidelberg, August 2010.

Reg05. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Sah99. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

ZY17. J. Zhang and Y. Yu. Two-round pake from approximate sph and instantiations from lattices. Cryptology ePrint Archive, Report 2017/838, 2017. To Appear in Asiacrypt 2017.

Appendix

A CCA2 and tag-CCA Security

In this section, we remind the definitions of IND-CCA2 and tag-IND-CCA2 encryption schemes, recall the generic transformation from the latter to the former, before proving tag-IND-CCA2 security for the scheme of Section 2.2.

A.1 Definitions

Definition A.1 (Labeled Encryption Scheme). *A (labeled) public-key encryption scheme is defined by four algorithms:*

– $\mathsf{KeyGen}(1^n)$ *takes as input a unary representation of the security parameter and generates a pair of keys* $(\mathsf{dk}, \mathsf{ek})$*, where* dk *is the secret decryption key and* ek *is the public encryption key;*
– $\mathsf{Encrypt}(\mathsf{ek}, \mathsf{label}, M; \rho)$ *produces a ciphertext* C *on the input message* M *under the label* label *and encryption key* ek*, using the random coins* ρ*;*
– $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{label}, C)$ *outputs the plaintext* M *encrypted in* C *under the label* label*, or* $\perp$*;*

and satisfies the following property:

– ***Correctness.*** *For any security parameter* n*, with overwhelming probability over* $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$*, for any label* label*, for any message* M*, for any ciphertext* $C \leftarrow \mathsf{Encrypt}(\mathsf{ek}, \mathsf{label}, M; \rho)$*, we have* $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{label}, C) = M$*.*

Definition A.2 (IND-CCA2 Security). *An encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* IND-CCA2 *if the advantage of any polynomial-time adversary* $\mathcal{A}$ *in distinguishing* $\mathsf{Exp}^{\mathsf{cca}-0}_{\mathcal{E},\mathcal{A}}(1^n)$ *from* $\mathsf{Exp}^{\mathsf{cca}-1}_{\mathcal{E},\mathcal{A}}(1^n)$ *is negligible in the security parameter* n*, where the experiments* $\mathsf{Exp}^{\mathsf{cca}-b}_{\mathcal{E},\mathcal{A}}(1^n)$ *are depicted in Fig. 3. Informally, this notion states that an adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and can ask several decryption of ciphertexts as long as they are not the challenge one.*

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

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

```
$\mathsf{Exp}^{\mathsf{cca}-b}_{\mathcal{E},\mathcal{A}}(1^n)$
1. $(\mathsf{dk},\mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$
2. $(\mathsf{label}^*, M_0, M_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\mathsf{dk},\cdot,\cdot)}(\mathsf{ek})$
3. $C^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, \mathsf{label}^*, M_b)$
4. $b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\mathsf{dk},\cdot,\cdot)}(\mathsf{st}, C^*)$
5. IF $(\mathsf{label}^*, C^*) \in \mathcal{CT}$ RETURN $0$
6. ELSE RETURN $b'$
7. Add $(\mathsf{label}, C)$ to $\mathcal{CT}$
8. RETURN $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{label}, C)$
```

Fig. 3: Security Experiment for CCA2 security.

– *The line* 5 *is replaced by:*
  6. IF $(\mathsf{label}^*, \cdot) \in \mathcal{CT}$ *RETURN 0.*
  In other words, the adversary is not allowed to query the decryption oracle on a ciphertext with the same label label *(also called a tag and denoted* u *in this context) as the challenge one.*
– *In addition the adversary chooses the label* label^* *before seeing* ek*, i.e., there is a line* 0*:*
  0. $(\mathsf{label}^*, \mathsf{st}_0) \xleftarrow{\$} \mathcal{A}(1^n)$
  and the line 2 *is replaced by:*
  2. $(M_0, M_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\mathsf{dk},\cdot,\cdot)}(\mathsf{st}_0, \mathsf{ek})$*.*

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

A.2 From Tag-IND-CCA2 to IND-CCA2

We can convert a tag-IND-CCA2 encryption scheme $(\mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ with message space $\{0,1\}$ and label (a.k.a., tag) space $\{0,1\}^n$ into an IND-CCA2 encryption scheme $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with message space $\{0,1\}^\nu$ (for some ν polynomial in n) and label space $\{0,1\}^*$, using [DDN03]. Concretely, we suppose that we have a strongly unforgeable one-time signature scheme and we define:

– $\mathsf{KeyGen}(1^n)$ outputs $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}'(1^n)$;
– $\mathsf{Encrypt}(\mathsf{ek}, \mathsf{label} \in \{0,1\}^*, M \in \{0,1\}^\nu)$ generates a signature key sk and an associated verification key pk (for the strongly unforgeable one-time signature, we suppose that pk can be represented as a n-bit string without loss of generality), computes for $1 \le i \le \nu$, $C_i \leftarrow \mathsf{Encrypt}'(\mathsf{ek}, \mathsf{pk}, M_i)$, and outputs $C := (C_1, \ldots, C_\nu, \mathsf{pk}, \sigma)$, where σ is a signature under sk of $(C_1, \ldots, C_\nu, \mathsf{pk}, \mathsf{label})$;

– $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{label} \in \{0,1\}^*, C)$ parses C as $(C_1, \ldots, C_\nu, \mathsf{pk}, \sigma)$, abort (i.e., return $\perp$) if σ is not a valid signature of $(C_1, \ldots, C_\nu, \mathsf{pk}, \mathsf{label})$ under pk, otherwise computes for $1 \le i \le \nu$, $M_i = \mathsf{Decrypt}'(\mathsf{dk}, \mathsf{pk}, C_i)$, and output the bit string $M \in \{0,1\}^\nu$ corresponding to the concatenation of $M_1, \ldots, M_\nu$.

A.3 Proof of Tag-IND-CCA2 Security of our Encryption Scheme (Theorem 2.9)

The proof follows closely the proof of the original scheme in [MP12]. We proceed with Hybrid games.

Hybrid H_0. The first hybrid game H_0 is the tag-IND-CCA2 game described in Fig. 3.

Hybrid H_1, Setup. In a second game H_1, we set the public key to be $\boldsymbol{A}_0 = [\bar{\boldsymbol{A}}\ ;\ \boldsymbol{R}\bar{\boldsymbol{A}} - \boldsymbol{G}\ h(u^*)]$, where $(\boldsymbol{T}, \boldsymbol{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, with $u^* = \mathsf{label}^*$ being the label chosen by the adversary for the challenge ciphertext, $\boldsymbol{T} = [-\boldsymbol{R} \,|\, \boldsymbol{I}]$, and $\boldsymbol{A} = [\bar{\boldsymbol{A}}\,;\, \boldsymbol{R}\bar{\boldsymbol{A}}]$. Note that $\boldsymbol{A}_0$ is statistically close to uniform, so that this new public key is statistically indistinguishable from the one from H_0.

Hybrid H_1, decryption queries. To handle decryption queries on tags $u \neq u^*$, the reduction simply outputs

$$\begin{cases} \mu & \text{if } g^{-1}_{\boldsymbol{A}_0}(\boldsymbol{T}, 2\boldsymbol{c}, h(u - u^*)) = 2\boldsymbol{e} + (0, \ldots, 0, \mu) \text{ where } \boldsymbol{e} \in \mathbb{Z}^m \\ & \quad \text{and } \|\boldsymbol{e}\| \le B' \text{ with } B' := q/\Theta(\sqrt{m}) \\ \perp & \text{otherwise.} \end{cases} ,$$

By the correctness of the $g^{-1}_{\boldsymbol{A}_0}$ algorithm (Lemma 2.8), this procedure outputs μ if and only if $d(\boldsymbol{c} - \mathsf{Encode}(\mu), \Lambda(\boldsymbol{A}_u)) < B'$, which is exactly the same behavior than in game H_0.

Hybrid H_1, challenge ciphertext. For the challenge ciphertext, choose $\mu \in \{0,1\}$, and set tag $u = u^*$. Choose $s \in \mathbb{Z}_q^n$, $\boldsymbol{e} \leftarrow D^m_{\mathbb{Z},\sigma}$, and set $\bar{\boldsymbol{b}} = \bar{\boldsymbol{A}}\boldsymbol{s} + \boldsymbol{e}$. Define $\boldsymbol{Q} = [\boldsymbol{I}_{\bar{m}}\ ;\ \boldsymbol{R}]$. Note that $\boldsymbol{Q}\bar{\boldsymbol{b}} = \boldsymbol{A}_{u^*}\boldsymbol{s} + \boldsymbol{Q}\boldsymbol{e}$. We then set the ciphertext to be:

$$\boldsymbol{c} = \boldsymbol{Q}\bar{\boldsymbol{b}} + \hat{\boldsymbol{e}} + \mathsf{Encode}(\mu) \ ,$$

where $\hat{\boldsymbol{e}} \leftarrow D^m_{\mathbb{Z},\sqrt{\Sigma}}$ and $\Sigma = t^2\boldsymbol{I}_m - \sigma^2\boldsymbol{Q}\boldsymbol{Q}^t$.[17] We note that

$$\boldsymbol{c} = \boldsymbol{A}_{u^*}\boldsymbol{s} + \boldsymbol{e}' + \mathsf{Encode}(\mu) \ , \quad \text{where } \boldsymbol{e}' = \boldsymbol{Q}\boldsymbol{e} + \hat{\boldsymbol{e}} \ .$$

We will argue that $\boldsymbol{c}$ is distributed as in game H_0. For this, it suffices to show that $\boldsymbol{e}'$ is negligibly close to $D_{\mathbb{Z}^m,t}$. Because $\boldsymbol{Q}\boldsymbol{e}$ belongs to $\mathbb{Z}^m$ the distribution $\boldsymbol{Q}\boldsymbol{e}' + D_{\mathbb{Z}^m,\sqrt{\Sigma}}$ of $\boldsymbol{e}'$ is equal to $D_{\mathbb{Z}^m,t,\boldsymbol{Q}\boldsymbol{e}}$. It remains to apply the convolution Theorem of Peikert [Pei10, Theorem 3.1], as already detailed in [MP12, Section 5.4].

[17] The procedure to sample from such a distribution is described in [Pei10, MP12].

Hybrid H_2. In a third game H_2, we only change the challenge ciphertext, and we now pick $\bar{\boldsymbol{b}}$ uniformly random in $\mathbb{Z}_q^{\bar{m}}$, which is indistinguishable from the previous game by the assumption of hardness of the $\mathrm{LWE}_{\chi,q}$ problem, for $\chi = D_{\mathbb{Z},\sigma}$.

In H_2, the adversary receives $\boldsymbol{c} = \boldsymbol{Q}\boldsymbol{b} + \hat{\boldsymbol{e}} + \mathsf{Encode}(\mu)$, with $\boldsymbol{Q}\bar{\boldsymbol{b}} = [\bar{\boldsymbol{b}}\,;\,\boldsymbol{R}\bar{\boldsymbol{b}}]^t$. But $(\bar{\boldsymbol{A}}, \boldsymbol{R}\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \boldsymbol{R}\bar{\boldsymbol{b}})$ is statistically negligibly close to uniform over the randomness of $\boldsymbol{R} \leftarrow D^{nk\times\bar{m}}$ by the leftover hash lemma. In particular, $\boldsymbol{c}$ is uniform and independent from the public key $\boldsymbol{A}_0$ and the message μ, so the advantage of the adversary is negligible in game H_2. $\square$

B SPHF

In this appendix, we formally define approximate word-independent SPHFs and describe the generic transformations of SPHFs sketched in Section 2.3 and summarized in Fig. 2.

B.1 Formal Definition of Approximate Word-Independent SPHF

Definition B.1. *An approximate word-independent SPHF is defined as in Definition 2.13 except that the algorithm* ProjKG *does not take as input the word* χ*, approximate correctness is modified accordingly, and smoothness is replaced by the following stronger property:*
Adaptive smoothness. *For any $n \in \mathbb{N}$, if* $(\mathsf{ltrap}, \mathsf{lpar}) \leftarrow \mathsf{Setup.lpar}(1^n)$*, with overwhelming probability over the randomness of* $\mathsf{Setup.lpar}$*, for all functions f onto $\mathcal{X} \setminus \mathscr{L}_{\mathsf{lpar}}$ the following distributions have statistical distance negligible in n:*

$$\left\{ (\mathsf{lpar}, f(\mathsf{hp}), \mathsf{hp}, \mathsf{H}) \;\middle|\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar}),\ \mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, f(\mathsf{hp})), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}) \end{array} \right\},$$

$$\left\{ (\mathsf{lpar}, f(\mathsf{hp}), \mathsf{hp}, \mathsf{H}) \;\middle|\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar}),\ \mathsf{H} \leftarrow \{0,1\}^{\nu}, \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}) \end{array} \right\}.$$

An approximate word-independent SPHF is called a word-independent SPHF if it is $\epsilon(n)$-correct with $\epsilon(n)$ negligible in the security parameter n.

B.2 Generic Transformations of Bit-PHFs and SPHFs

From Approximate Bit-PHF to Approximate SPHF. This transformation is straightforward, we simply need to increase the size of the output of the hash function, by sampling several independent hash keys hk, and concatenating the output of all the corresponding Hash results.

Lemma B.2. *Let* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *be an ϵ-correct approximate bit-PHF. Then the SPHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *defined as follows is an $(\epsilon + \epsilon')$-correct approximate SPHF, for any constant $\epsilon' > 0$.*

- $\mathsf{HashKG}(\mathsf{lpar})$ *generates a hashing key* $\mathsf{hk} = (\mathsf{hk}_1, \ldots, \mathsf{hk}_\nu)$ *by running ν times* $\mathsf{HashKG}'(\mathsf{lpar})$*, where $\nu = \Omega(n)$;*

– $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *derives a projection key* hp *from the hashing key* hk*, by computing* $\mathsf{hp}_i = \mathsf{ProjKG}'(\mathsf{hk}_i, \mathsf{lpar}, \chi)$ *(for* $i \in \{1, \dots, \nu\}$*) and setting* $\mathsf{hp} = (\mathsf{hp}_1, \dots, \mathsf{hp}_\nu)$.
– $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs a hash value* $\mathsf{H} \in \{0,1\}^\nu$*, by computing the various hash values* $\mathsf{H}_i = \mathsf{Hash}(\mathsf{hk}_i, \mathsf{lpar}, \chi)$ *(for* $i \in \{1, \dots, \nu\}$*) and concatenating the ouputs:* $\mathsf{H} = \mathsf{H}_1 \| \dots \| \mathsf{H}_\nu$*;*
– $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *outputs a projected hash value* $\mathsf{pH} \in \{0,1\}^\nu$*, by computing the projected hash values* $\mathsf{pH}_i = \mathsf{ProjHash}'(\mathsf{hp}_i, \mathsf{lpar}, \chi, w)$ *(for* $i \in \{1, \dots, \nu\}$*) and concatenating them:* $\mathsf{pH} = \mathsf{pH}_1 \| \dots \| \mathsf{pH}_\nu$*;*

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

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

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

From Approximate Correctness to Correctness. There exists a generic transformation, implicit in [KV09], from an approximate SPHF to an SPHF. The idea is quite simple, it requires the use of an error correcting code (noted ECC in the following) capable of correcting an ϵ-fraction of errors.

Lemma B.3. *Let* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *be an* ϵ*-correct approximate SPHF (with hash values in* $\{0,1\}^\nu$*) and* ECC *be an error correcting code capable of correcting an* ϵ*-fraction of errors, and with Hlen-bit codewords. Then the SPHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *defined as follows is a (regular) SPHF:*

– $\mathsf{HashKG}(\mathsf{lpar})$ *sets* $\mathsf{hk}_1 \leftarrow \mathsf{HashKG}'(\mathsf{lpar})$*, and picks a random value* hk_2 *from the input set of* ECC*. It then returns* $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$*;*
– $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *computes* $\mathsf{hp}_1 \leftarrow \mathsf{ProjKG}'(\mathsf{hk}_1, \mathsf{lpar}, \chi)$*, and computes* $c = \mathsf{ECC}(\mathsf{hk}_2), \mathsf{H}' \leftarrow \mathsf{Hash}'(\mathsf{hk}_1, \mathsf{lpar}, \chi)$*, and sets* $\mathsf{hp}_2 = c \oplus \mathsf{H}'$*;*
– $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *simply outputs* $\mathsf{H} = \mathsf{hk}_2$*;*
– $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *computes* $\mathsf{pH}' = \mathsf{ProjHash}'(\mathsf{hp}_1, \mathsf{lpar}, \chi, w)$ *and sets* $\mathsf{pH} = \mathsf{ECC}^{-1}(\mathsf{pH}' \oplus \mathsf{hp}_2)$.

We stress that this transformation always gives a SPHF (but not a word-independent SPHF), even if the original approximate SPHF is an approximate word-independent SPHF, as the ProjKG algorithm requires to run the approximate Hash' algorithm, and therefore requires the knowledge of the word χ.

Proof. **Approximate-correctness.** In an honest execution, the approximate correctness guarantees that $\mathsf{HW}(\mathsf{pH}', \mathsf{H}') \leq \epsilon \cdot n$. In particular, this means that $\mathsf{HW}(\mathsf{pH}' \oplus \mathsf{hp}_2, c) \leq \epsilon \cdot n$. Now, the capacity of the error-correcting code leads to the conclusion: $\mathsf{pH} = \mathsf{H}$.

Smoothness. Smoothness of the original SPHF ensures that when $\chi \notin \mathscr{L}$, H' is negligibly close to uniform even when knowing hp'. Therefore, it completely masks c (in hp_2) and thus $\mathsf{H} = \mathsf{hk}_2$ is negligibly close to uniform even when knowing $\mathsf{hp}_1 = \mathsf{hp}'$ and $\mathsf{hp}_2 = c \oplus \mathsf{H}'$. □

From Imperfectly Universal Word-Independent Bit-PHFs to Word-Independent SPHFs. The idea is quite simple: we first XOR the hash values of several independent executions of the word-independent bit-PHF to amplify universality and get a statistically universal word-independent bit-PHF. To convert the resulting word-independent bit-PHF into a word-independent SPHF, we then increase the output length using basic concatenation and parallel executions as in Lemma B.2.

Lemma B.4. *Let* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *be a* ϵ*-universal word-independent bit-PHF. Then the SPHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *defined as follows is a word-independent SPHF:*

– $\mathsf{HashKG}(\mathsf{lpar})$ *generates a hashing key* $\mathsf{hk} = (\mathsf{hk}_{(1,1)}, \dots, \mathsf{hk}_{(\eta,\nu)})$ *by running* $\eta \cdot \nu$ *times the original hashing key generation* $\mathsf{HashKG}'(\mathsf{lpar})$*, where* $\eta = \omega(-\log n / \log \epsilon)$ *and* ν *is the output length of the SPHF;*
– $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$ *derives a projection key* hp *from the hashing key* hk*, by computing* $\mathsf{hp}_{(i,j)} = \mathsf{ProjKG}'(\mathsf{hk}_{(i,j)}, \mathsf{lpar})$ *and setting* $\mathsf{hp} = (\mathsf{hp}_{(1,1)}, \dots, \mathsf{hp}_{(\eta,\nu)})$.
– $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs a hash value* $\mathsf{H} \in \{0,1\}^\nu$*, by computing the various hash values* $\mathsf{H}_{(i,j)} = \mathsf{Hash}(\mathsf{hk}_{(i,j)}, \mathsf{lpar}, \chi)$*, and then* $\mathsf{H}_j = \mathsf{H}_{1,j} \oplus \dots \oplus \mathsf{H}_{\eta,j}$ *(for* $i \in \{1, \dots, \eta\}$*,* $j \in \{1, \dots, \nu\}$*), and concatenating the ouputs:* $\mathsf{H} = \mathsf{H}_1 \| \dots \| \mathsf{H}_\nu$*;*
– $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *outputs a projected hash value* $\mathsf{pH} \in \{0,1\}^\nu$*, by computing the projected hash values* $\mathsf{pH}_{(i,j)} = \mathsf{ProjHash}(\mathsf{hk}_{(i,j)}, \mathsf{lpar}, \chi)$*, and then* $\mathsf{pH}_j = \mathsf{pH}_{1,j} \oplus \dots \oplus \mathsf{pH}_{\eta,j}$ *(for* $i \in \{1, \dots, \eta\}$*,* $j \in \{1, \dots, \nu\}$*), and concatenating the ouputs:* $\mathsf{pH} = \mathsf{pH}_1 \| \dots \| \mathsf{pH}_\nu$*;*

Proof. **Correctness.** Correctness is straightforward as the original word-independent bit-PHF is statistically correct.
Smoothness. With overwhelming probability over lpar, for each $j \in \{1, \dots, \nu\}$, for any projection key hp, we have:

$$\begin{aligned}
&\left| 2 \cdot \Pr_{\mathsf{hk}}\left[\mathsf{H}_{1,j} \oplus \dots \oplus \mathsf{H}_{\eta,j} = 1 \;\middle|\; \forall i \in \{1, \dots, \eta\},\ \mathsf{hp}_{i,j} = \mathsf{ProjKG}(\mathsf{hk}_{i,j}, \mathsf{lpar})\right] - 1 \right| \\
&= \left| \mathop{\mathbb{E}}_{\mathsf{hk}}\left[(-1)^{\mathsf{H}_{1,j}} \cdots (-1)^{\mathsf{H}_{\eta,j}} \;\middle|\; \forall i \in \{1, \dots, \eta\},\ \mathsf{hp}_{i,j} = \mathsf{ProjKG}(\mathsf{hk}_{i,j}, \mathsf{lpar})\right] \right| \\
&= \left| \mathop{\mathbb{E}}_{\mathsf{hk}_{1,j}}\left[(-1)^{\mathsf{H}_{1,j}} \;\middle|\; \mathsf{hp}_{1,j} = \mathsf{ProjKG}(\mathsf{hk}_{1,j}, \mathsf{lpar})\right] \cdots \right. \\
&\qquad\qquad \left. \cdots \mathop{\mathbb{E}}_{\mathsf{hk}_{\eta,j}}\left[(-1)^{\mathsf{H}_{\eta,j}} \;\middle|\; \mathsf{hp}_{\eta,j} = \mathsf{ProjKG}(\mathsf{hk}_{\eta,j}, \mathsf{lpar})\right] \right| \\
&\le \epsilon^\eta = 2^{-\omega(\log n)} \quad ,
\end{aligned}$$

where $\mathsf{H}_{(i,j)} = \mathsf{Hash}(\mathsf{hk}_{(i,j)}, \mathsf{lpar}, \chi)$, $\mathbb{E}$ denotes the expectation, and the second equality comes from the independence of the hashing keys $\mathsf{hk}_{i,j}$. In other words, if $\nu = 1$, then we would have constructed a statistically universal word-independent bit-PHF.

Smoothness follows immediately. □

C Additional Content for Applications

We now give a little more details on the various possibles applications of our new SPHFs.

C.1 One-Round PAKE

As explained in Section 5.1, we can construct a 1-round PAKE using our word-independent SPHF from Section 4 by following the first instantiation in [KV11]. More precisely the construction uses the framework in [KV11]. This framework requires an IND-CCA2 encryption scheme with an associated word-independent SPHF. Unfortunately, we only know how to construct a word-independent SPHF for an IND-CPA encryption scheme over lattices. So, as in [KV11, ZY17], we use the Naor-Yung paradigm [NY90, Sah99]: an IND-CCA2 ciphertext consists of two IND-CPA ciphertexts (under two different keys) and a simulation-sound zero-knowledge (SS-NIZK) proof proving that both ciphertexts encrypt the same value. Decryption of such a ciphertext consists in decrypting the first IND-CPA ciphertext and checking the SS-NIZK. We already have an SPHF able to check the first part (Section 4), while the second part can be publicly checked. Thus we have the primitives required to use the framework of [KV11], with a very minor difference: the gap between smoothness and correctness. But as in [KV09], this is not an issue.

It remains to construct an SS-NIZK for the language of pairs of IND-CPA ciphertexts à la Micciancio-Peikert (Section 2.2) which encrypts the same value. We could use any generic construction.

But if we want a more efficient one, we can follow [ZY17]. Let us suppose first that the plaintext is just one bit. In that case, we can use the SS-NIZK defined in [ZY17, Section 5.3], where the matrix U is defined as

$$U = \begin{pmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & \lceil q/2 \rceil \end{pmatrix} .$$

and $\boldsymbol{w}$ is just the bit message. We recall that this SS-NIZK is based on applying the Fiat-Shamir transform [FS87] to an efficient three-round public-coin honest-verifier zero-knowledge proof with quasi-unique responses.

For multi-bit messages, messages can be encrypted bit-by-bit twice using the IND-CPA encryption scheme. The SS-NIZK needs to prove that each pair of ciphertexts encrypt the same value. The corresponding efficient three-round public-coin protocols can be run in parallel. The resulting protocol is a three-round public-coin honest-verifier zero-knowledge proof with quasi-unique responses for the language we are interested in. And its Fiat-Shamir transform is an SS-NIZK for this language.

C.2 Honest-Verifier Zero-Knowledge

Generic construction. Following the methodology from [BCPW15], using our SPHF in Section 3, we can construct honest-verifier zero-knowledge proofs for any NP language of the form $\ddot{\mathscr{L}} = \{\ddot{\chi} \mid \exists \ddot{w}, \ddot{\mathscr{R}}(\ddot{\chi}, \ddot{w})\}$ where $\ddot{\mathscr{R}}$ is a polynomial-size circuit.

For the sake of simplicity, we suppose that all gates of the circuit $\ddot{\mathscr{R}}$ are NAND gates. We suppose that we have an IND-CPA encryption scheme and an SPHF for the languages $\widetilde{\mathscr{L}} \subseteq \mathscr{L}$ of ciphertexts C_1, C_2, C_3 encrypting values (b_1, b_2, b_3) so that $b_3 = \mathrm{NAND}(b_1, b_2)$, such that $\widetilde{\mathscr{L}}$ is the set of encryptions of b_i that fits the NAND gate evaluation, while $\mathscr{L}$ is the set of ciphertexts whose decryptions fit the gate evaluation.

Our participants are going to interact in three rounds. First the prover commits to every single wires. Then for each gate, the verifier computes a projection key for an SPHF associated with a valid evaluation of the NAND gate, and sends the corresponding projection keys, while keeping the XOR of the hash values. Finally, the prover using his witnesses evaluates the various projective hash values, and sends the XOR to the verifier that can then compare it with its own evaluation. The scheme is described in Fig. 4.

Common reference string: Encryption key ek for an IND-CPA encryption scheme.

Wire commitments: For each wire i in the circuit $\ddot{\mathscr{R}}$ evaluated on the word $\ddot{\chi}$ for the argument and a witness $\ddot{w}$, the prover is going to encrypto its value b in $C_i \leftarrow \mathsf{Encrypt}(\mathsf{ek}, b)$, and keeps the corresponding randomness (or witness) $\rho_i = w_i$. He then sends the ciphertexts $C = \{C_i\}_i$.

Verifying the gates:

1. For each NAND gate j, linking the wires i_1, i_2 to i_3, the verifier computes $\mathsf{hk}_j, \mathsf{hp}_j, \mathsf{H}_j$ for the SPHF described in the text and the word $\chi = (C_{i_1}, C_{i,2}, C_{i_3})$. The verifier then sends $\mathsf{hp} = \{\mathsf{hp}_j\}_j$.
2. For each hp_j, the prover using w_{i_1}, w_{i_2} and w_{i_3} can now recover pH_j, he then computes $\mathsf{pH} = \bigoplus_j \mathsf{pH}_j$ and sends it to the prover.

Validation:

– The verifier computes $\mathsf{H} = \bigoplus_j \mathsf{H}_j$, and accepts if $\mathsf{H} = \mathsf{pH}$.

Fig. 4: Honest-verifier zero-knowledge argument from SPHFs

Completeness comes directly from the correctness of the underlying SPHFs, while soundness comes from their smoothness. A simulator (for the honest-verifier zero-knowledge property) would encrypt dummy values, and compute pH using the hashing keys hk_j. Under the IND-CPA security of the encryption scheme, this simulation is computationally indistinguishable from the real experiment, ensuring that the previous construction is indeed honest-verifier zero-knowledge.

Instantiation. It remains to show how to construct SPHFs for the above language $\widetilde{\mathscr{L}} \subseteq \mathscr{L}$ from our SPHF in Section 3, when the IND-CPA encryption scheme is our IND-CCA2 encryption scheme à la Micciancio-Peikert of Section 2.2.

We remark that a set of wire values (b_1, b_2, b_3) corresponds to a valid evaluation of a NAND gate ($b_3 = \mathrm{NAND}(b_1, b_2)$) if and only if $(b_1 = 0 \wedge b_2 = 0 \wedge b_3 = 1) \vee (b_1 = 0 \wedge b_2 = 1 \wedge b_3 = 1) \vee (b_1 = 1 \wedge b_2 = 0 \wedge b_3 = 1) \vee (b_1 = 1 \wedge b_2 = 1 \wedge b_3 = 0)$. Therefore, we can express similarly $\widetilde{\mathscr{L}}/\mathscr{L}$ in term of the languages $\widetilde{\mathscr{L}_{i,b}}/\mathscr{L}_{i,b}$ (defined as the language of tuples of ciphertexts $(C_j)_j$, such that the ciphertext C_i encrypts / decrypts to b) for $i \in \{i_1, i_2, i_3\}$. These languages can be handled by the SPHF presented in Section 3.

It is therefore sufficient to show how to combine SPHFs for the languages $\widetilde{\mathscr{L}_{i,b}} \subseteq \mathscr{L}_{i,b}$ to get an SPHF for the language $\widetilde{\mathscr{L}} \subseteq \mathscr{L}$. For that we can use the techniques introduced in [ACP09] to handle combinations of SPHFs (for conjunctions "$\cap$" and disjunctions "$\cup$").

Combination of SPHF We recall, and transpose the constructions described in [ACP09].

Conjunctions and disjunctions of SPHFs. We assume to be given two smooth projective hash functions SPHF_1 and SPHF_2, on the sets corresponding to the languages $\widetilde{\mathscr{L}_{\mathsf{lpar}_1}}$ and $\widetilde{\mathscr{L}_{\mathsf{lpar}_2}}$: $\mathrm{SPHF}_i = (\mathsf{HashKG}_i, \mathsf{ProjKG}_i, \mathsf{Hash}_i, \mathsf{ProjHash}_i)$.

For a given $\chi \in \mathcal{X}$, we naturally define $\mathsf{hk}_1, \mathsf{hk}_2, \mathsf{hp}_1, \mathsf{hp}_2$ as before.

A smooth projective hash system for the language $\widetilde{\mathscr{L}} = \widetilde{\mathscr{L}_{\mathsf{lpar}_1}} \cap \widetilde{\mathscr{L}_{\mathsf{lpar}_2}}$ is then defined as follows, if $\chi \in \widetilde{\mathscr{L}_{\mathsf{lpar}_1}} \cap \widetilde{\mathscr{L}_{\mathsf{lpar}_2}}$ and w_i is a witness that $\chi \in \widetilde{\mathscr{L}_{\mathsf{lpar}_i}}$, for both $i = 1, 2$:

- $\mathsf{HashKG}_{\widetilde{\mathscr{L}_{\mathsf{lpar}}}}(\mathsf{lpar}) = \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$;
- $\mathsf{ProjKG}_{\widetilde{\mathscr{L}_{\mathsf{lpar}}}}(\mathsf{hk}, \mathsf{lpar}, \chi) = \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$;
- $\mathsf{Hash}_{\widetilde{\mathscr{L}_{\mathsf{lpar}}}}(\mathsf{hk}, \mathsf{lpar}, \chi) = \mathsf{Hash}_1(\mathsf{hk}_1, \mathsf{lpar}_1, \chi) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, \mathsf{lpar}_2, \chi)$;
- $\mathsf{ProjHash}_{\widetilde{\mathscr{L}_{\mathsf{lpar}}}}(\mathsf{hp}, \mathsf{lpar}, \chi, (w_1, w_2)) =$
  $\mathsf{ProjHash}_1(\mathsf{hp}_1, \mathsf{lpar}_1, \chi, w_1) \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2, \mathsf{lpar}_2, \chi, w_2)$.

Smoothness is then guaranteed for words outside $\mathscr{L} = \mathscr{L}_{\mathsf{lpar}_1,\mathsf{ltrap}_1} \cap \mathscr{L}_{\mathsf{lpar}_2,\mathsf{ltrap}_2}$.

Similarly, a smooth projective hash system for the language $\widetilde{\mathscr{L}} = \widetilde{\mathscr{L}_{\mathsf{lpar}_1}} \cup \widetilde{\mathscr{L}_{\mathsf{lpar}_2}}$ is defined as follows, if $\chi \in \widetilde{\mathscr{L}_{\mathsf{lpar}_1}} \cup \widetilde{\mathscr{L}_{\mathsf{lpar}_2}}$ and w is a witness that χ belongs to one of the languages.

- $\mathsf{HashKG}_{\widetilde{\mathscr{L}}_{\mathsf{lpar}}}(\mathsf{lpar}) = \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$;
- $\mathsf{ProjKG}_{\widetilde{\mathscr{L}}_{\mathsf{lpar}}}(\mathsf{hk}, \mathsf{lpar}, \chi) = \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta)$
  where $\mathsf{hp}_\Delta = \mathsf{Hash}_1(\mathsf{hk}_1, \mathsf{lpar}_1, \chi) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, \mathsf{lpar}_2, \chi)$
- $\mathsf{Hash}_{\widetilde{\mathscr{L}}_{\mathsf{lpar}}}(\mathsf{hk}, \mathsf{lpar}, \chi) = \mathsf{Hash}_1(\mathsf{hk}_1, \mathsf{lpar}_1, \chi)$;
- $\mathsf{ProjHash}_{\widetilde{\mathscr{L}}_{\mathsf{lpar}}}(\mathsf{hp}, \mathsf{lpar}, \chi, w) =$
$$\begin{cases} \mathsf{ProjHash}_1(\mathsf{hp}_1, \mathsf{lpar}_1, \chi, w) & \text{if } \chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar}_1} \ , \\ \mathsf{hp}_\Delta \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2, \mathsf{lpar}_2, \chi, w) & \text{if } \chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar}_2} \ . \end{cases}$$

Once again, smoothness is then guaranteed for words outside $\mathscr{L} = \mathscr{L}_{\mathsf{lpar}_1, \mathsf{ltrap}_1} \cup \mathscr{L}_{\mathsf{lpar}_2, \mathsf{ltrap}_2}$.

C.3 Witness Encryption

Another application would be in the domain of witness encryption [GGSW13] for statements derived from the language of ciphertexts as defined in Example 2.12.[18]

Definition C.1. *Let* $(\widetilde{\mathscr{L}}_{\mathsf{lpar}} \subseteq \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}} \subseteq \mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar},\mathsf{ltrap}}$ *be languages defined as before. A witness encryption scheme for these languages is defined by the two probabilistic polynomial-time algorithms:* $(\mathsf{Encrypt}_{\mathsf{WE}}, \mathsf{Decrypt}_{\mathsf{WE}})$*, where:*

- $\mathsf{Encrypt}_{\mathsf{WE}}(1^n, \chi, M)$ *generates a ciphertext* C *from a plaintext* M*, a security parameter* n*, and a word* $\chi \in \mathcal{X}$*.*
- $\mathsf{Decrypt}_{\mathsf{WE}}(C, w)$ *decrypts the ciphertext* C *into* M *using the witness.*

It has to satisfy the two following properties:

- ***Correctness.*** *For any security parameter* n*, message* $M \in \{0,1\}$*, and* $\chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar}}$ *such that* $\widetilde{\mathscr{R}}(\chi, w)$ *holds, we have*
$$\Pr[\mathsf{Decrypt}_{\mathsf{WE}}(\mathsf{Encrypt}_{\mathsf{WE}}(1^n, \chi, M), w) = M] \geq 1 - \mathrm{negl}(n) \ .$$
- ***Soundness.*** *For any probabilistic polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* $\mathrm{negl}(.)$ *such that for any* $n \in \mathbb{N}$*, if* $(\mathsf{ltrap}, \mathsf{lpar}) \leftarrow \mathsf{Setup.lpar}(1^n)$*, with overwhelming probability over the randomness of* $\mathsf{Setup.lpar}$*, for any* $\chi \notin \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}}$*:*
$$\Pr_{\mathcal{A}}[\mathsf{Encrypt}_{\mathsf{WE}}(1^n, \chi, 0) = 1] - \Pr_{\mathcal{A}}[\mathsf{Encrypt}_{\mathsf{WE}}(1^n, \chi, 1) = 1] < \mathrm{negl}(n) \ .$$

In the original definition [GGSW13], there was a voluntary gap between the soundness and correctness, as nothing is said for words in the language with no known witnesses. Over lattice-based schemes, it is natural to extend the gap, by considering $\widetilde{\mathscr{L}}_{\mathsf{lpar}}$ for the correctness, while defining the soundness for $\mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}}$. Another minor difference is the introduction of language parameters $(\mathsf{ltrap}, \mathsf{lpar})$,

[18] The concept of using SPHF to generically build Witness Encryption was already mentioned as folklore in the introduction of [ABP15a], but as far as we know it was not properly detailed anywhere.

as we are considering only restricted languages (and not NP-complete languages as in [GGSW13]). We point out that our construction achieves statistical soundness (i.e., against any adversary) and therefore also satisfies (up to this additional gap and the language parameters) adaptive soundness as defined in [BH15].

Concretely, here is our construction. Assuming an SPHF on the language $\widetilde{\mathscr{L}}_{\mathsf{lpar}}$, we can build a witness encryption as follows:

– $\mathsf{Encrypt}_{\mathsf{WE}}(1^n, \chi, M)$ outputs $C = (\mathsf{hp}, \mathsf{H} \oplus M)$, by running $\mathsf{HashKG}(\mathsf{lpar})$, $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$, $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ to compute $\mathsf{hk}, \mathsf{hp}, \mathsf{H}$.
– $\mathsf{Decrypt}_{\mathsf{WE}}(C, w)$ recovers $M = P \oplus \mathsf{pH}$ by parsing C as $C = (\mathsf{hp}, P)$, and computing $\mathsf{pH} = \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$.

Theorem C.2. *The above construction is a correct and statistically sound witness encryption scheme.*

Proof. Under the correctness of the underlying SPHF, one obtains:

$$\Pr\left[\mathsf{Decrypt}_{\mathsf{WE}}(\mathsf{Encrypt}_{\mathsf{WE}}(1^n, \chi, M), w) = M\right] \geq 1 - \mathrm{negl}(n) \enspace .$$

It is interesting to note, that in case of an ϵ-approximate SPHF, one can still achieve an ϵ-approximate correctness for the encryption.

The smoothness of the SPHF, ensures that for χ not in the language, H is seemingly random from the point of view of an adversary, hence $\mathsf{H} \oplus M$ is too, which guarantees the desired soundness. $\square$