Disjunctions for Hash Proof Systems: New Constructions and Applications

Michel Abdalla, Fabrice Benhamouda, and David Pointcheval

ENS, Paris, France [⋆]

Abstract. Hash Proof Systems were first introduced by Cramer and Shoup (Eurocrypt'02) as a tool to construct efficient chosen-ciphertext-secure encryption schemes. Since then, they have found many other applications, including password authenticated key exchange, oblivious transfer, and zero-knowledge arguments. One of the aspects that makes hash proof systems so interesting and powerful is that they can be seen as implicit proofs of membership for certain languages. As a result, by extending the family of languages that they can handle, one often obtains new applications or new ways to understand existing schemes. In this paper, we show how to construct hash proof systems for the disjunction of languages defined generically over cyclic, bilinear, and multilinear groups. Among other applications, this enables us to construct the most efficient one-time simulation-sound (quasi-adaptive) non-interactive zero-knowledge arguments for linear languages over cyclic groups, the first one-round group password-authenticated key exchange without random oracles, the most efficient threshold structure-preserving chosen-ciphertext-secure encryption scheme, and the most efficient one-round password authenticated key exchange in the UC framework.

Keywords. Hash Proof System, Non-Interactive Zero-Knowledge Proof, Group Password Authenticated Key Exchange, Threshold Encryption, Linearly Homomorphic Signature, Structure Preserving Primitive.

1 Introduction

Hash Proof Systems or Smooth Projective Hash Functions (SPHFs), which can be seen as a kind of implicit designated-verifier proofs of membership [4, 7], were originally introduced by Cramer and Shoup [12] as a way to build efficient chosen-ciphertext-secure (IND-CCA) encryption schemes. Informally speaking, SPHFs are families of pairs of functions $(\mathsf{Hash}, \mathsf{ProjHash})$ defined on a language $\mathscr{L} \subset \mathcal{X}$. These functions are indexed by a pair of associated keys $(\mathsf{hk}, \mathsf{hp})$, where the hashing key hk and the projection key hp can be seen as the private and public keys, respectively. When computed on a word $C \in \mathscr{L}$, both functions should lead to the same result: $\mathsf{Hash}(\mathsf{hk}, \mathscr{L}, C)$ with the hashing key and $\mathsf{ProjHash}(\mathsf{hp}, \mathscr{L}, C, w)$ with the projection key and a witness w that $C \in \mathscr{L}$. Of course, if $C \notin \mathscr{L}$, such a witness does not exist, and the smoothness property states that $\mathsf{Hash}(\mathsf{hk}, \mathscr{L}, C)$ is independent of hp. As a consequence, the value $\mathsf{Hash}(\mathsf{hk}, \mathscr{L}, C)$ cannot be guessed even with the knowledge of hp.

[⋆] CNRS – UMR 8548, INRIA, and PSL

Since their introduction, SPHFs have been used in various applications, including Password Authenticated Key Exchange (PAKE) [16, 23, 24], Oblivious Transfer [1, 22], One-Time Relatively-Sound Non-Interactive Zero-Knowledge Arguments [19], Zero-Knowledge Arguments [6], and Trapdoor Smooth Projective Hash Functions (TSPHFs) [6]. An SPHF for a language $\mathscr{L}$ also directly leads to a witness encryption scheme [15] for the same language $\mathscr{L}$: encrypting a message m for a word C consists in generating an hashing key hk and a projection key hp and outputting hp together with m masked with the hash value $\mathsf{Hash}(\mathsf{hk}, \mathscr{L}, C)$ of C under hk. If we know a witness w for C, we can compute this hash value from hp, while if $C \notin \mathscr{L}$, this hash value statistically masks the message.

As explained in [6], various variants of SPHFs have been proposed over the years, depending on whether the projection key hp is allowed to depend on C and whether the smoothness holds even when C is chosen after having seen hp. For witness encryption, for example, the weakest notion (hp depends on C) is sufficient, while for encryption schemes and one-round PAKE, the strongest notion (hp does not depend on C and C may be chosen after hp in the smoothness property) is required. In this article, we focus on the strongest notion of SPHF, also called KV-SPHF in [6], since it has more applications. However, most parts of the paper could be adapted to use the weaker GL-SPHF notion.

Expressiveness of SPHFs. Due to the wide range of applications of SPHFs, one may wonder what kind of languages can be handled by SPHFs. First, since SPHF implies statistical witness encryption, it is important to remark that it is impossible to construct SPHF for any NP language, unless the polynomial hierarchy collapses [15]. Nevertheless, as the many different applications show, the class of languages supported by SPHFs can be very rich.

Diverse Groups and Diverse Vector Spaces. In [12], Cramer and Shoup showed that SPHFs can handle any language based on what they call a diverse group. Most, if not all, constructions of SPHF are based on diverse groups. However, in the context of languages over cyclic groups, bilinear groups or even multilinear groups, diverse groups may appear slightly too generic. That is why, in [6], Benhamouda et al. introduced a generic framework (later called *diverse vector space*) encompassing most known SPHFs based over these kinds of groups. It can be seen as particular diverse groups with more mathematical structure, namely using vector spaces instead of groups. In this article, we are mainly interested on SPHFs based on diverse vector spaces.

Operations on SPHFs. In order to enrich the class of languages that can be handled by SPHFs, Abdalla, Chevalier, and Pointcheval [4] showed how to build SPHFs for languages that can be described in terms of disjunctions and conjunctions of simpler languages for which SPHFs are known to exist. Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two such languages. In the particular case of conjunctions, when given SPHFs for $\mathscr{L}_1$ and $\mathscr{L}_2$, they showed how to build an SPHF for the conjunction $\mathscr{L} = \mathscr{L}_1 \times \mathscr{L}_2$, so that a word $C = (C_1, C_2) \in \mathscr{L}$ if and only if $C_1 \in \mathscr{L}_1$ and $C_2 \in \mathscr{L}_2$. Note that this definition is a generalization of the "classical" conjunction: $C_1 \in \mathscr{L}$ if and only if $C_1 \in \mathscr{L}_1$ and $C_1 \in \mathscr{L}_2$, which we can get by setting $C_1 = C_2$.

In the case of disjunctions, when given SPHFs for $\mathscr{L}_1$ and $\mathscr{L}_2$, Abdalla et al. showed how to build an SPHF for language $\mathscr{L} = (\mathscr{L}_1 \times \mathcal{X}_2) \cup (\mathcal{X}_1 \times \mathscr{L}_2)$, so that $C = (C_1, C_2) \in \mathscr{L}$ if and only if $C_1 \in \mathscr{L}_1$ or $C_2 \in \mathscr{L}_2$. In particular, a witness for $C = (C_1, C_2) \in \mathscr{L}$ can be either a witness w_1 for $C_1 \in \mathscr{L}_1$ or a witness w_2 for $C_2 \in \mathscr{L}_2$. As for conjunctions, by setting $C_1 = C_2$, one gets the "classical" disjunction: $C = (C_1, C_1) \in \mathscr{L}$ if and only if $C_1 \in \mathscr{L}_1$ or $C_1 \in \mathscr{L}_2$.

Unfortunately, while the conjunction of two strong SPHFs in [4] yields a strong SPHF, the same is not true for disjunctions, where the projection key hp necessarily depends on C. And this greatly limits its applications[1].

1.1 Results

Disjunction of SPHFs. Our first main result is to show how to construct the disjunction of two SPHFs for two languages based on diverse vector spaces. Essentially, the only requirement for the construction is that it is possible to compute a pairing between an element of the first language $\mathscr{L}_1$ and an element of the second language $\mathscr{L}_2$. Concretely, if we have a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of some prime order p (we say that $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a bilinear group), and if $\mathscr{L}_1$ is defined over $\mathbb{G}_1$ and $\mathscr{L}_2$ over $\mathbb{G}_2$, then our construction provides an SPHF for the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$. Furthermore, this disjunction can be repeated multiple times, if multilinear maps are available. The only limitation is that the complexity of our constructions grows exponentially with the number of repetitions, therefore limiting the total number of disjunctions that we can compute.

Application: Constant-Size NIZK and One-Time Simulation-Sound NIZK. First, we show how to use disjunctions of SPHFs to create efficient *non-interactive zero-knowledge arguments (NIZK)* and even *one-time simulation-sound NIZK*, i.e., NIZK in which a dishonest (polynomial-time) prover cannot produce a valid proof of a false statement, even when seeing one simulated proof on a statement of its choice (which may be false). The proof size consists of only two group elements, even for the one-time simulation-sound version, assuming the language we are interested in can be handled by an SPHF over some group $\mathbb{G}_1$, where $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is an asymmetric bilinear group, and assuming DDH is hard in $\mathbb{G}_2$. The languages handled roughly consist of languages defined by "linear" equations over $\mathbb{G}_1$, such as the DDH language, the language of valid Cramer-Shoup [11] ciphertexts and many other useful languages as shown in [6, 20].

Our NIZK is slightly different from a usual NIZK, since the common reference string depends on the language. Jutla and Roy called them quasi-adaptive NIZK in [20], and showed that they can replace NIZK in several applications.

Our one-time simulation-sound NIZK yields a very efficient structure-preserving threshold $\mathsf{IND\text{-}CCA}$ encryption scheme, with the shortest ciphertext size so far. Threshold means the decryption key can be shared between parties and a ciphertext can be

[1] A reader familiar with [17] may wonder why the methods in [17] cannot be applied to provide a form of disjunction, given that SPHFs exist for languages of quadratic pairing equations over commitments [6]. Unfortunately, this technique would not yield a real SPHF, since additional commitments would be required.

decrypted if and only if enough parties provide a partial decryption of it using their key share, while structure-preserving means it can be used in particular with Groth-Sahai NIZK [18] or our new NIZK construction. In addition, this new encryption can be used in the one-round password authenticated key exchange (PAKE) scheme in the UC model in [6] to obtain an improvement of up to 30% in the communication complexity, under the same assumptions.

Other Applications. Another important application is the first *one-round group password authenticated key exchange (GPAKE)* with n players, assuming the existence of a $(n-1)$-multilinear map and the hardness of the n-linear assumption n-Lin without random oracles[2]. This was an open problem. We remark, however, that our construction only works for small values of n since the overall complexity of the protocol and the gap in the security reduction grows exponentially in n. We note, however, that the tripartite PAKE which only requires pairings is reasonably efficient since it consists of flows with 61 group elements for each user (5 for the Cramer-Shoup ciphertext and 56 for the projection key).

A second application is a new construction for TSPHF, which supports slightly more languages than the original one, but which is slightly less efficient. A TSPHF (Trapdoor Smooth Projective Hash Function [6]) is a variant of an SPHF with a full-fledged zero-knowledge flavor: there exists a trapdoor for computing the hash value of any word $C \in \mathscr{X}$ when only given C and the projection key hp.

Finally, the unforgeability of the one-time linearly homomorphic structure-preserving signature scheme of Libert et al. [25] can be explained by the smoothness of some underlying SPHF, which can be seen as the disjunction of two SPHFs. This new way of seeing their signature scheme directly shows how to extend it to other assumptions, such as SXDH, κ-Lin, or even any MDDH assumption [13] secure in bilinear groups.

Pseudo-Random Projective Hash Functions (PrPHFs) and More Efficient Applications. For our NIZK and our new TSPHF, the construction essentially consists in the disjunction of an SPHF for the language in which we are interested, and another SPHF for a language which is used to provide extra features (zero-knowledge and "public verifiability" for our NIZK and trapdoor for our TSPHF). This second language $\mathscr{L}_2$ is supposed to be a hard subset membership one, i.e., it is hard to distinguish a random word $C_2 \in \mathscr{L}_2$ from a random word $C_2 \in \mathscr{X}_2 \setminus \mathscr{L}_2$.

To get more efficient applications, we introduce the notion of pseudo-random projective hash functions (PrPHFs) which are particular SPHFs over trivial languages, i.e., languages $\mathscr{L} = \mathscr{X}$, where all words are in the language. Of course, smoothness becomes trivial, in this case. That is why PrPHFs are supposed to have another property called *pseudo-randomness*, which ensures that if the parameters of the language $\mathscr{L}$ and the word C are chosen at random, given a projection key hp (and no witness for C), the hash value H of C appears random.

[2] At the time the first version of this paper was made public [2], the multilinear map construction by Coron et al. [10] seemed to be a plausible candidate. However, as recently shown by Cheon et al. [9], this is no longer the case. Unfortunately, no current candidate multilinear map construction is known to work for our framework for $n \geq 3$.

We then show that we can replace the second hard subset membership language in our NIZK and our TSPHF by a trivial language with a PrPHF, assuming a certain property over the first language $\mathscr{L}_1$ (which is almost always verified). This conversion yields slightly shorter proofs (for our NIZK and our one-time simulation-sound NIZK) or slightly shorter projection keys (for our TSPHF).

Related Work. Until now, the most efficient NIZK for similar languages was the one of Jutla and Roy [21], and the most efficient *one-time* simulation-sound NIZK was the *unbounded* simulation-sound NIZK of Libert et al. [26]. Even though all these constructions have constant-size proofs, our second NIZK is slightly more efficient for κ-linear assumptions, with $\kappa \geq 2$, while our one-time simulation-sound NIZK is about ten times shorter. Moreover, our construction might be simpler to understand due to its modularity. We provide a detailed comparison in Section 7.3.

1.2 Organization

In the next section, we give the high level intuition for all our constructions and their applications. Then, after recalling some preliminaries in Section 3, we give the details of our construction of disjunctions of SPHFs in Section 4, which is one of our main contributions. We then show how to build efficient NIZK and one-time simulation-sound NIZK from it in Section 5. After that, we introduce the notion of PrPHF in Section 6 and show in Section 7 how this can improve some of our previous applications. These last two sections are much more technical: although the underlying ideas are similar to the ones in previous sections, the proofs are more complex. Due to lack of space, details of our two other applications, namely one-round GPAKE and TSPHF, are presented in the full version, but an overview is available in Section 2.3.

2 Overview of Our Constructions

2.1 Disjunction of Languages

Intuition. From a very high point of view, the generic framework [6] enables us to construct an SPHF for any language $\mathscr{L}$ which is a subspace of the vector space of all words $\mathcal{X}$.

It is therefore possible to do the conjunction of two languages $\mathscr{L}_1$ and $\mathscr{L}_2$ supported by this generic framework by remarking that $\mathscr{L}_1 \times \mathscr{L}_2$ is a subspace of the vector space $\mathcal{X}_1 \times \mathcal{X}_2$. This construction of conjunctions is an “algebraic” version of the conjunction proposed in [4].

Unfortunately, the same approach cannot be directly applied to the case of disjunctions, because $(\mathscr{L}_1 \times \mathcal{X}_2) \cup (\mathcal{X}_1 \times \mathscr{L}_2)$ is not a subspace of $\mathcal{X}_1 \times \mathcal{X}_2$, and the subspace generated by the former union of sets is $\mathcal{X}_1 \times \mathcal{X}_2$. In this article, we solve this issue by observing that, instead of using $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$, we can consider the tensor product of $\mathcal{X}_1$ and $\mathcal{X}_2$: $\mathcal{X} = \mathcal{X}_1 \otimes \mathcal{X}_2$. Then the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$ can be seen as the subspace $\mathscr{L}$ of $\mathcal{X}$ generated by: $\mathscr{L}_1 \otimes \mathcal{X}_2$ and $\mathcal{X}_1 \otimes \mathscr{L}_2$. Notice that $(\mathscr{L}_1 \otimes \mathcal{X}_2) \cup (\mathcal{X}_1 \otimes \mathscr{L}_2)$ is not a subspace and so $\mathscr{L}$ is much larger than this union of sets. But we can prove that if $C_1 \otimes C_2 \in \mathscr{L}$, then $C_1 \in \mathscr{L}_1$ or $C_2 \in \mathscr{L}_2$.

Before providing more details about these constructions, let us first briefly recall the main ideas of the generic framework for constructing SPHFs.

Generic Framework for SPHFs. The generic framework for SPHFs in [6] uses a common formalization for cyclic groups, bilinear groups, and even multilinear groups[3] (of prime order p), called graded rings[4].

Basically, graded rings enable us to use a ring structure over these groups: the addition and the multiplication of two elements u and v, denoted $u + v$ and $u \bullet v$, respectively, correspond to the addition and the multiplication of their discrete logarithms. For example, if g is a generator of a cyclic group $\mathbb{G}$, and a and b are two scalars in $\mathbb{Z}_p$, $a + b = a + b$, $a \bullet b = a \cdot b$ (because the "discrete logarithm" of a scalar is the scalar itself), $g^a + g^b = g^{a+b}$, and $g^a \bullet g^b = g_T^{a \cdot b}$, with g_T a generator of another cyclic group $\mathbb{G}_T$ of order p.

Of course, computing $g^a \bullet g^b = g_T^{a \cdot b}$ requires a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, if the discrete logarithms of g^a and g^b are not known. And if such a bilinear map exists, we can compute $g^a \bullet g^b$ as $e(g^a, g^b)$. For a similar reason, the multiplication of three group elements via $\bullet$ would require a trilinear map. Therefore, graded rings can be seen as the ring $\mathbb{Z}_p$ with some limitations on the multiplication. Here, to avoid technicalities, the group of each element is implicit, and we suppose that above constraints on the multiplications are satisfied. Formal details are left to the following sections.

From a high level point of view, in this framework, we suppose there exists a map θ from the set of words $\mathscr{X}$ to a vector space $\hat{\mathscr{X}}$ of dimension n, together with a subspace $\hat{\mathscr{L}}$ of $\hat{\mathscr{X}}$, generated by a family of vectors $(\boldsymbol{\Gamma_i})_{i=1}^k$, such that $C \in \mathscr{L}$ if and only if $\theta(C) \in \hat{\mathscr{L}}$. When the function θ is clear from context, we often write $\hat{\boldsymbol{C}} \coloneqq \theta(C)$.

A witness for a word $C \in \mathscr{L}$ is a vector $\boldsymbol{\lambda} = (\lambda_i)_{i=1}^k$ so that $\hat{\boldsymbol{C}} = \theta(C) = \sum_{i=1}^k \lambda_i \bullet \boldsymbol{\Gamma_i}$. In other words, it consists of the coefficients of a linear combination of $(\boldsymbol{\Gamma_i})_{i=1}^k$ equal to $\hat{\boldsymbol{C}}$.

Then, a hashing key hk is just a random linear form $\mathsf{hk} \coloneqq \alpha \in \hat{\mathscr{X}}^*$ ($\hat{\mathscr{X}}^*$ being the dual vector space of $\hat{\mathscr{X}}$, i.e., the vector space of linear maps from $\hat{\mathscr{X}}$ to $\mathbb{Z}_p$), and the associated projection key is the vector of its values on $\boldsymbol{\Gamma_1}, \dots, \boldsymbol{\Gamma_k}$:

$$\mathsf{hp} \coloneqq \boldsymbol{\gamma} = (\gamma_i)_{i=1}^k = (\alpha(\boldsymbol{\Gamma_i}))_{i=1}^k.$$

The hash value of a word C is then $H \coloneqq \alpha(\hat{\boldsymbol{C}})$. If $\boldsymbol{\lambda}$ is a witness for $C \in \mathscr{L}$, then the latter can also be computed as:

$$H = \alpha(\hat{\boldsymbol{C}}) = \alpha\left(\sum_{i=1}^k \lambda_i \bullet \boldsymbol{\Gamma_i}\right) = \sum_{i=1}^k \lambda_i \bullet \alpha(\boldsymbol{\Gamma_i}) = \sum_{i=1}^k \lambda_i \bullet \gamma_i,$$

which only depends on the witness $\boldsymbol{\lambda}$ and the projection key hp. The smoothness comes from the fact that, if $C \notin \mathscr{L}$, then $\hat{\boldsymbol{C}} \notin \hat{\mathscr{L}}$ and $\hat{\boldsymbol{C}}$ is linearly independent from $(\boldsymbol{\Gamma_i})_{i=1}^k$. Hence, $\alpha(\hat{\boldsymbol{C}})$ looks random even given $\mathsf{hp} = (\alpha(\boldsymbol{\Gamma_i}))_{i=1}^k$.

[3] In this work, we need a multilinear map for which DDH, κ-Lin, or any MDDH assumption [13] hold in the multilinear groups. Unfortunately, as explained in Footnote 2, no current candidate multilinear map construction is known to work for our framework.

[4] Graded rings were named after graded encodings systems [14] and are unrelated to the mathematical notion of graded rings.

For a reader familiar with [12], the generic framework is similar to a diverse group, but with more structure: a vector space instead of a simple group. When θ is the identity function, $(\mathcal{X}^*, \mathcal{X}, \mathscr{L}, \mathbb{Z}_p)$ is a diverse group. We remark, however, that one does not need to know diverse groups to understand our paper.

Example 1 (SPHF for DDH). Let us illustrate this framework for the DDH language: let g, h be two generators of a cyclic group $\mathbb{G}$ of prime order p, let $\mathcal{X} = \mathbb{G}^2$ and $\mathscr{L} = \{(g^r, h^r)^\intercal \in \mathcal{X} \mid r \in \mathbb{Z}_p\}$. We set $\hat{\mathcal{X}} = \mathcal{X}$, $\hat{\mathscr{L}} = \mathscr{L}$ and θ the identify function so that $C = \hat{\boldsymbol{C}} = (u, v)^\intercal$. $\hat{\mathscr{L}}$ is generated by the column vector $\boldsymbol{\Gamma_1} = (g, h)^\intercal$. The witness for $C = (g^r, h^r)^\intercal$ is $\lambda_1 = r$. The hashing key $\mathsf{hk} = \alpha \xleftarrow{\$} \hat{\mathcal{X}}^*$ can be represented by a row vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2) \in \mathbb{Z}_p^{1\times 2}$ and

$$\mathsf{hp} = \gamma_1 = \alpha(\boldsymbol{\Gamma_1}) = \boldsymbol{\alpha} \bullet \boldsymbol{\Gamma_1} = g^{\alpha_1} \cdot h^{\alpha_2}$$
$$H = \alpha(\hat{\boldsymbol{C}}) = \boldsymbol{\alpha} \bullet \hat{\boldsymbol{C}} = u^{\alpha_1} \cdot v^{\alpha_2} = \gamma_1 \bullet r = \gamma_1^r.$$

This is exactly the original SPHF of Cramer and Shoup for the DDH language in [12].

Remark on the Notation of Vectors (Transposition) and Link with [13]. Compared to [6], in this paper, we transposed all the vectors and matrices: elements of $\mathcal{X}$ are now column vectors, while hashing keys (elements of $\mathcal{X}^*$) are row vectors. This seems more natural and makes our notation closer to the one of Escala et al. [13].

Warm up: Conjunction of Languages. As a warm up, let us first construct the conjunction $\mathscr{L} = \mathscr{L}_1 \times \mathscr{L}_2$ of two languages $\mathscr{L}_1 \subset \mathcal{X}_1$ and $\mathscr{L}_2 \subset \mathcal{X}_2$ supported by the generic framework, in a more algebraic way than the one in [4]. We can just set:

$$\hat{\mathcal{X}} := \hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2 \qquad n := n_1 + n_2$$
$$\hat{\mathscr{L}} := \hat{\mathscr{L}}_1 \times \hat{\mathscr{L}}_2 \qquad k := k_1 + k_2$$
$$\theta((C_1, C_2)) = \hat{\boldsymbol{C}} := \begin{pmatrix} \theta_1(C_1) \\ \theta_2(C_2) \end{pmatrix} \qquad (\boldsymbol{\Gamma_i})_{i=1}^k := \left(\begin{pmatrix} \boldsymbol{\Gamma_i^{(1)}} \\ \boldsymbol{0} \end{pmatrix}_{i=1}^{k_1}, \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\Gamma_i^{(2)}} \end{pmatrix}_{i=1}^{k_2} \right)$$

This is what is implicitly done in all conjunctions of SPHFs in [6], for example.

Example 2 (SPHF for Conjunction of DDH). Let g_1, h_1, g_2, h_2 be four generators of a cyclic group $\mathbb{G}$ of prime order p. Let $\mathcal{X}_1 = \mathcal{X}_2 = \mathbb{G}^2$ and $\mathscr{L}_i = \{(g_i^{r_i}, h_i^{r_i})^\intercal \in \mathcal{X}_i \mid r_i \in \mathbb{Z}_p\}$ for $i = 1, 2$. We set $\hat{\mathcal{X}}_i = \mathcal{X}_i$, $\hat{\mathscr{L}}_i = \mathscr{L}_i$ and θ_i the identify function so that $C_i = \hat{\boldsymbol{C}}_i = (u_i, v_i)^\intercal$, for $i = 1, 2$. $\hat{\mathscr{L}}_i$ is generated by the column vector $\boldsymbol{\Gamma_1^{(i)}} = (g_i, h_i)^\intercal$. The witness for $C_i = (g_i^{r_i}, h_i^{r_i})^\intercal$ is $\lambda_1^{(i)} = r_i$. Then, the SPHF for the conjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$ is defined by:

$$\hat{\mathcal{X}} := \hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2 = \mathbb{G}^4 \qquad n = 4 \quad k = 2$$
$$\hat{\mathscr{L}} := \hat{\mathscr{L}}_1 \times \hat{\mathscr{L}}_2 = \{(g_1^{r_1}, h_1^{r_1}, g_2^{r_2}, h_2^{r_2})^\intercal \mid r_1, r_2 \in \mathbb{Z}_p\}$$
$$\boldsymbol{\Gamma_1} := (g_1, h_1, 1, 1)^\intercal \in \mathbb{G}^4 \qquad \boldsymbol{\Gamma_2} := (1, 1, g_2, h_2)^\intercal \in \mathbb{G}^4$$
$$\theta(C) := \hat{\boldsymbol{C}} := (u_1, v_1, u_2, v_2)^\intercal \in \mathbb{G}^4 \text{ for } C = (C_1, C_2) = ((u_1, v_1)^\intercal, (u_2, v_2)^\intercal)$$

The hashing key $\mathsf{hk} = \alpha \overset{\$}{\leftarrow} \hat{\mathcal{X}}^*$ can be represented by a row vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in \mathbb{Z}_p^{1\times 4}$ and

$$\mathsf{hp} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha} \bullet \boldsymbol{\Gamma_1} \\ \boldsymbol{\alpha} \bullet \boldsymbol{\Gamma_2} \end{pmatrix} = \begin{pmatrix} g_1^{\alpha_1} \cdot h_1^{\alpha_2} \\ g_2^{\alpha_3} \cdot h_2^{\alpha_4} \end{pmatrix}$$

$$H = \alpha(\hat{\boldsymbol{C}}) = \boldsymbol{\alpha} \bullet \hat{\boldsymbol{C}} = u_1^{\alpha_1} \cdot v_1^{\alpha_2} \cdot u_2^{\alpha_3} \cdot v_2^{\alpha_4} = \gamma_1 \bullet r_1 + \gamma_2 \bullet r_2 = \gamma_1^{r_1} \cdot \gamma_2^{r_2}.$$

Disjunction of Languages. We first remark we cannot naively extend the previous construction by choosing $\hat{\mathcal{X}} = \hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2$ and $\hat{\mathscr{L}} = (\hat{\mathscr{L}}_1 \times \hat{\mathcal{X}}_2) \cup (\hat{\mathcal{X}}_1 \times \hat{\mathscr{L}}_2)$, because, in this case $\hat{\mathscr{L}}$ is not a subspace, and the subspace generated by $\hat{\mathscr{L}}$ is $\hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2$. That is why we use tensor products of vector spaces instead of direct product of vector spaces. Concretely, we set

$$\begin{aligned} \hat{\mathcal{X}} &\coloneqq \hat{\mathcal{X}}_1 \otimes \hat{\mathcal{X}}_2 & n &\coloneqq n_1 n_2 \\ \hat{\mathscr{L}} &\coloneqq \langle (\hat{\mathscr{L}}_1 \otimes \hat{\mathcal{X}}_2) \cup (\hat{\mathcal{X}}_1 \otimes \hat{\mathscr{L}}_2) \rangle & k &\coloneqq k_1 n_2 + n_1 k_2 \\ \theta(C) = \hat{\boldsymbol{C}} &\coloneqq \hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2 \end{aligned}$$

where the notation $\langle V \rangle$ is the vector space generated by V. The vectors $\boldsymbol{\Gamma_i}$ are described in detail in the core of the paper. This construction works since, if $\hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2 \in \hat{\mathscr{L}}$ then, thanks to properties of the tensor product, $\hat{\boldsymbol{C}}_1 \in \hat{\mathscr{L}}_1$ or $\hat{\boldsymbol{C}}_2 \in \hat{\mathscr{L}}_2$.

It is important to remark that computing a tensor product implies computing a multiplication. So if $\hat{\boldsymbol{C}}_1$ in $\hat{\mathcal{X}}_1$ and $\hat{\boldsymbol{C}}_2$ in $\hat{\mathcal{X}}_2$ are over some cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$, we need a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ to actually be able to compute $\hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2$. More generally, doing the disjunction of K languages over cyclic groups requires a K-way multilinear map. This can be seen in the following example and we formally deal with this technicality in the core of the paper.

Example 3 (SPHF for Disjunction of DDH). Let us use the same notation as in Example 2, except that this time $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is an asymmetric bilinear group (e is a bilinear map: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$), g_1, h_1 are generators of $\mathbb{G}_1$, g_2, h_2 are generators of $\mathbb{G}_2$, and $\mathcal{X}_i = \hat{\mathcal{X}}_i = \mathbb{G}_i^2$ (instead of $\mathbb{G}^2$) for $i = 1, 2$.

The disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$ is defined by

$$\begin{aligned} \hat{\mathcal{X}} &\coloneqq \hat{\mathcal{X}}_1 \otimes \hat{\mathcal{X}}_2 = \mathbb{G}_T^4 & n &\coloneqq 4 \\ \hat{\mathscr{L}} &\coloneqq \langle (\hat{\mathscr{L}}_1 \otimes \hat{\mathcal{X}}_2) \cup (\hat{\mathcal{X}}_1 \otimes \hat{\mathscr{L}}_2) \rangle & k &\coloneqq 4 \end{aligned}$$

$$\boldsymbol{\Gamma_1} \coloneqq \begin{pmatrix} g_1 \\ h_1 \end{pmatrix} \otimes \begin{pmatrix} 1 \in \mathbb{Z}_p \\ 0 \in \mathbb{Z}_p \end{pmatrix} = \begin{pmatrix} g_1^1 \\ g_1^0 \\ h_1^1 \\ h_1^0 \end{pmatrix} = \begin{pmatrix} g_1 \\ 1 \\ h_1 \\ 1 \end{pmatrix} \in \mathbb{G}_1^4$$

$$\boldsymbol{\Gamma_2} \coloneqq \begin{pmatrix} g_1 \\ h_1 \end{pmatrix} \otimes \begin{pmatrix} 0 \in \mathbb{Z}_p \\ 1 \in \mathbb{Z}_p \end{pmatrix} = \begin{pmatrix} g_1^0 \\ g_1^1 \\ h_1^0 \\ h_1^1 \end{pmatrix} = \begin{pmatrix} 1 \\ g_1 \\ 1 \\ h_1 \end{pmatrix} \in \mathbb{G}_1^4$$

$$\boldsymbol{\Gamma_3} \coloneqq \begin{pmatrix} 1 \in \mathbb{Z}_p \\ 0 \in \mathbb{Z}_p \end{pmatrix} \otimes \begin{pmatrix} g_2 \\ h_2 \end{pmatrix} = \begin{pmatrix} g_2^1 \\ h_2^1 \\ g_2^0 \\ h_2^0 \end{pmatrix} = \begin{pmatrix} g_2 \\ h_2 \\ 1 \\ 1 \end{pmatrix} \in \mathbb{G}_2^4$$

$$\boldsymbol{\Gamma_4} \coloneqq \begin{pmatrix} 0 \in \mathbb{Z}_p \\ 1 \in \mathbb{Z}_p \end{pmatrix} \otimes \begin{pmatrix} g_2 \\ h_2 \end{pmatrix} = \begin{pmatrix} g_2^0 \\ h_2^0 \\ g_2^1 \\ h_2^1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ g_2 \\ h_2 \end{pmatrix} \in \mathbb{G}_2^4$$

$$\begin{aligned}\theta(C) = \hat{\boldsymbol{C}} \coloneqq \hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2 &= (u_1 \bullet u_2, u_1 \bullet v_2, v_1 \bullet u_2, v_1 \bullet v_2)^\intercal \\ &= (e(u_1,u_2), e(u_1,v_2), e(v_1,u_2), e(v_1,v_2))^\intercal \in \mathbb{G}_T^4,\end{aligned}$$

for $C = (C_1, C_2) = ((u_1, v_1), (u_2, v_2))$. The generating family of $\hat{\mathscr{L}}$ we used here is $(\boldsymbol{\Gamma_1}, \boldsymbol{\Gamma_2}, \boldsymbol{\Gamma_3}, \boldsymbol{\Gamma_4})$. As seen after, if we know the witness r_1 for C_1, we can use $\boldsymbol{\Gamma_1}$ and $\boldsymbol{\Gamma_2}$ to compute the hash value of $C = (C_1, C_2)$, while if we know the witness r_2 for C_2, we can use $\boldsymbol{\Gamma_3}$ and $\boldsymbol{\Gamma_4}$ to compute the hash value of C. Obviously this generating family is not free, since $\hat{\mathscr{L}}$ has dimension 3 and this family has cardinality 4.

The witnesses $\boldsymbol{\lambda}$ for a word $C = (C_1, C_2)$ are

$$\begin{cases} (r_1 \bullet u_2, r_1 \bullet v_2, 0, 0) & \text{if } (u_1, v_1) = (g^{r_1}, h^{r_1}) \quad \text{(i.e., if } r_1 \text{ is a witness for } C_1) \\ (0, 0, r_2 \bullet u_1, r_2 \bullet v_1) & \text{if } (u_2, v_2) = (g^{r_2}, h^{r_2}) \quad \text{(i.e., if } r_2 \text{ is a witness for } C_2), \end{cases}$$

the hashing key $\mathsf{hk} = \alpha \overset{\$}{\leftarrow} \hat{\mathcal{X}}^*$ can be represented by a row vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in \mathbb{Z}_p^{1 \times 4}$ and

$$\begin{aligned} \mathsf{hp} &= (\gamma_1, \gamma_2, \gamma_3, \gamma_4)^\intercal = (g_1^{\alpha_1} \cdot h_1^{\alpha_3},\ g_1^{\alpha_2} \cdot h_1^{\alpha_4},\ g_2^{\alpha_1} \cdot h_2^{\alpha_2},\ g_2^{\alpha_3} \cdot h_2^{\alpha_4})^\intercal \in \mathbb{G}_1^2 \times \mathbb{G}_2^2 \\ H &= \alpha(\hat{\boldsymbol{C}}) = \hat{\boldsymbol{C}} \bullet \boldsymbol{\alpha} = e(u_1,u_2)^{\alpha_1} \cdot e(u_1,v_2)^{\alpha_2} \cdot e(v_1,u_2)^{\alpha_3} \cdot e(v_1,v_2)^{\alpha_4} \\ &= \begin{cases} r_1 \bullet u_2 \bullet \gamma_1 + r_1 \bullet v_2 \bullet \gamma_2 = e(\gamma_1, u_2)^{r_1} e(\gamma_2, v_2)^{r_1}, \text{ if } (u_1, v_1) = (g_1^{r_1}, h_1^{r_1}) \\ r_2 \bullet u_1 \bullet \gamma_3 + r_2 \bullet v_1 \bullet \gamma_4 = e(u_1, \gamma_3)^{r_2} e(v_1, \gamma_4)^{r_2}, \text{ if } (u_2, v_2) = (g_2^{r_2}, h_2^{r_2}) \end{cases} \end{aligned}$$

The last equalities, which show the way the projection hashing works, explain the choice of the generating family $(\boldsymbol{\Gamma_i})_i$.

2.2 Main Application: One-Time Simulation-Sound NIZK Arguments

The language of the NIZK is $\mathscr{L}_1$, while $\mathscr{L}_2$ is a hard subset membership language used to build the NIZK. For the sake of simplicity, we suppose that $\mathscr{L}_2 = \hat{\mathscr{L}}_2$, $\mathcal{X}_2 = \hat{\mathcal{X}}_2$, and θ_2 is the identity function. We will consider the SPHF of the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$, so we need to suppose that it is possible to build it. For this high level overview, let us just suppose that $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a bilinear group and that $\mathscr{L}_1$ is defined over $\mathbb{G}_1$, $\mathscr{L}_2$ over $\mathbb{G}_2$. If DDH holds in $\mathbb{G}_2$, $\mathscr{L}_2$ can just be the DDH language in $\mathbb{G}_2$ recalled in Example 1.

The common reference string is a projection key hp for the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$, while the trapdoor (to simulate proofs) is the hashing key. Essentially, a proof

$\boldsymbol{\pi} = (\pi_{i_2})_{i_2}$ for a statement C_1 is just a vector of the hash values of $(C_1, \boldsymbol{e_{2,i_2}})$ where $(\boldsymbol{e_{2,i_2}})_{i_2}$ are the scalar vectors of the canonical base of $\hat{\mathscr{X}}_2$. These hash values are $\pi_{i_2} = \alpha(\hat{\boldsymbol{C}}_1 \otimes \boldsymbol{e_{2,i_2}})$, and can also be computed from the projection key hp and a witness for $\hat{\boldsymbol{C}}_1$.

The basic idea is that a valid proof for a word $C_1 \in \mathscr{L}_1$ enables us to compute the hash value H' of (C_1, C_2) for any word $C_2 \in \hat{\mathscr{X}}_2$, by linearly combining elements of the proof, since any word C_2 can be written as a linear combination of $(\boldsymbol{e_{2,i_2}})_{i_2}$:

$$H' := \sum_{i_2} \pi_{i_2} \bullet C_{2,i_2} = \sum_{i_2} \alpha(\hat{\boldsymbol{C}}_1 \otimes (C_{2,i_2} \bullet \boldsymbol{e_{2,i_2}})) = \alpha(\hat{\boldsymbol{C}}_1 \otimes C_2),$$

if $C_2 = \sum_{i_2} C_{2,i_2} \bullet \boldsymbol{e_{1,i_2}}$. Hence, for any word $C_2 \in \mathscr{L}_2$ for which we know a witness, we can compute the hash value of (C_1, C_2), either using a valid proof for C_1 (as H' above), or directly using the witness of C_2 and the projection key hp (as for any SPHF for a disjunction).

To check a proof, we basically check that for any word $C_2 \in \mathscr{L}_2$, these two ways of computing the hash value of (C_1, C_2) yields the same result. Thanks to the linearity of the language $\mathscr{L}_2$, it is sufficient to make this test for a family of words C_2 which generate $\mathscr{L}_2$, such as the words $\boldsymbol{\Gamma_j^{(2)}}$ (for $j = 1, \ldots, k_2$). We recall that the witness for $\boldsymbol{\Gamma_j^{(2)}}$ is the column vector $(0, \ldots, 0, 1, 0, \ldots, 0)^\intercal \in \mathbb{Z}_p^{k_2}$, where the j-th coordinate is 1.

The trapdoor, i.e., the hashing key, clearly enables us to simulate any proof, and the resulting proofs are perfectly indistinguishable from normal ones, hence the perfect zero-knowledge property. Moreover, the soundness comes from the fact that a proof for a word $C_1 \notin \mathscr{L}_1$ can be used to break the hard subset membership in $\mathscr{L}_2$.

More precisely, let us consider a soundness adversary which takes as input the projection key hp and which outputs a word $C_1 \notin \mathscr{L}_1$ and a valid proof π for C_1. On the one hand, such a valid proof enables us to compute the hash value H' of (C_1, C_2) for any word $C_2 \in \mathscr{L}_2$, by linearly combining elements of the proofs (as seen above), and the validity of the proof ensures the resulting value H' is correct if $C_2 \in \mathscr{L}_2$. On the other hand, we can also compute a hash value H of (C_1, C_2) for any $C_2 \in \mathscr{X}_2$ using the hashing key hk. Then, if $C_2 \in \mathscr{L}_2$, necessarily $H = H'$, while if $C_2 \notin \mathscr{L}_2$, the smoothness ensures that H looks completely random when given only hp. Since H' does not depend on hk but only on hp, it is different from H with overwhelming probability. Therefore, we can use such an adversary to solve the hard subset membership problem in $\mathscr{L}_2$ (namely, the DDH in $\mathbb{G}_2$ in the example below).

Example 4 (NIZK for DDH in $\mathbb{G}_1$, assuming DDH in $\mathbb{G}_2$). Using the SPHF in Example 3, the proof for a word $C_1 = (u_1 = g_1^r, v_1 = h_1^r) \in \mathbb{G}_1^2$ is the vector $\boldsymbol{\pi} = (\pi_1, \pi_2) \in \mathbb{G}_1^2$ where: π_1 is the hash value of $(C_1, (1,0)^\intercal) \in \mathbb{G}_1^2 \times \mathbb{Z}_p^2$ and π_2 is the hash value of $(C_1, (0,1)^\intercal) \in \mathbb{G}_1^2 \times \mathbb{Z}_p^2$. Concretely we have:

$$\pi_1 = \gamma_1 \bullet r = \gamma_1^r \in \mathbb{G}_1 \qquad\qquad \pi_2 = \gamma_2 \bullet r = \gamma_2^r \in \mathbb{G}_1.$$

This proof is valid if and only if:

$$e(\pi_1, g_2) \cdot e(\pi_2, h_2) = \pi_1 \bullet g_2 + \pi_2 \bullet h_2 \stackrel{?}{=} u_1 \bullet \gamma_3 + v_1 \bullet \gamma_4 = e(u_1, \gamma_3) \cdot e(v_1, \gamma_4).$$

This check can be done using the common reference string $\mathsf{hp} = (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$.

Finally, to simulate a proof for $C_1 = (u_1, v_1)$ without knowing any witness for C_1 but knowing the trapdoor $\mathsf{hk} = \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in \mathbb{Z}_p^{1\times 4}$, we compute π_1 and π_2 as follows:

$$\pi_1 \coloneqq u_1 \bullet \alpha_1 + v_1 \bullet \alpha_3 = u_1^{\alpha_1} \cdot v_1^{\alpha_3} \qquad \pi_2 \coloneqq u_1 \bullet \alpha_2 + v_1 \bullet \alpha_4 = u_1^{\alpha_2} \cdot v_1^{\alpha_4}.$$

To get a one-time simulation-sound NIZK, we replace the SPHF over $\mathscr{L}_1$ by a stronger kind of SPHF for which, roughly speaking, the hash value of a word $C \notin \mathscr{L}_1$ appears random even if we are given the projection key hp and the hash value of another word $C \in \mathscr{X}_1$ of our choice. We show that it is always possible to transform a normal SPHF into this stronger variant, assuming the existence of collision-resistant hash functions[5].

2.3 Other Applications

TSPHF. A TSPHF is an extension of an SPHF, with an additional CRS and an associated trapdoor, where the latter provides a way to efficiently compute the hash value of any word C knowing only the projection key hp. Since hp now needs to contain enough information to compute the hash value of any word in $\mathscr{X}$, the smoothness property of TSPHFs is no longer statistical but computational. As shown in [6], TSPHFs can be used to construct two-round zero-knowledge protocols and the most efficient one-round PAKE in the standard model.

TSPHF is a direct application of disjunctions of SPHFs: as for NIZK, the language we are interested in is $\mathscr{L}_1$, while $\mathscr{L}_2$ is a hard subset membership language. The common reference string contains a word $C_2 \in \mathscr{L}_2$, and the trapdoor is just a witness w_2 for this word. The hash value of some $C_1 \in \mathscr{X}_1$, is the hash value of (C_1, C_2) for the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$, which can be computed in two or three ways: using hk, or using hp and w_1 (classical projection hashing — possible only when $C_1 \in \mathscr{L}_1$ and w_1 is a witness for it), or using hp and w_2 (trapdoor). The smoothness comes from the hard subset membership property of $\mathscr{L}_2$ (which says that this common reference string is indistinguishable from a word $C_2 \notin \mathscr{L}_2$) and the fact that when $C_2 \notin \mathscr{L}_2$, the hash value of (C_1, C_2) appears random by smoothness when $C_1 \notin \mathscr{L}_1$, given only hp.

The resulting TSPHF is slightly less efficient than the construction in [6]: if $\mathscr{L}_2$ corresponds to the DDH language (Example 1), the projection key contains less than twice more elements than the original construction. But it has the advantage of handling more languages, since contrary to the original construction, there is no need to have a trapdoor $\mathscr{T}_{\mathsf{crs}}$ for crs which enables us to compute the discrete logarithms of all entries of Γ_1 (a property called witness-samplability in [20])[6].

[5] Actually, the use of collision-resistant hash functions could be avoided, but that would make the construction much less efficient.

[6] However, due to the definition of computational smoothness of TSPHF in [6], it is still required to have such a trapdoor $\mathscr{T}_{\mathsf{crs}}$ enabling to check whether a word C_1 is in $\mathscr{L}_1$ or not. It may be possible to change definitions to avoid that, but in all applications we are aware of, this is never a problem.

One-Time Linearly Homomorphic Structure-Preserving Signature. We can obtain the one-time linearly homomorphic structure-preserving signature scheme of messages in $\mathbb{G}_1^{n_1}$ of Libert et al. [25] and extend it to work under any hard-subset membership language assumption, such as the DDH language in Example 1 but also DLin or any MDDH assumption [13] as seen later (instead of just DLin as in the original paper). The construction is very similar to our NIZK construction.

Let $\mathscr{L}_2 = \hat{\mathscr{L}}_2 \subset \mathscr{X}_2 = \hat{\mathscr{X}}_2$ be a hard membership language and $\mathscr{X}_1 = \hat{\mathscr{X}}_1 = \mathbb{G}_1^{n_1}$ (the language $\mathscr{L}_1 = \hat{\mathscr{L}}_1$ will be defined later). The secret key is the hashing key $\mathsf{hk} = \alpha$ of the SPHF of the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$ (notice that it does not depend on the language $\hat{\mathscr{L}}_1$ but only on $\hat{\mathscr{X}}_1$), while the public key is the associated projection key when $\mathscr{L}_1 = \hat{\mathscr{L}}_1 = \{0\}$. The signature of a message $\boldsymbol{M} \in \hat{\mathscr{X}}_1 = \mathbb{G}_1^{n_1}$ is the vector of the hash values of $(\boldsymbol{M}, \boldsymbol{e_{2,i_2}})$ where $(\boldsymbol{e_{2,i_2}})_{i_2}$ are the scalar vectors of the canonical base of $\hat{\mathscr{X}}_2$. It can be computed using the secret key hk. Actually, this corresponds to the NIZK proof of $\boldsymbol{M}$ (computed using the trapdoor hk), in our NIZK scheme above. Checking the signature can be done by checking the validity of the proof using the projection key hp when $\hat{\mathscr{L}}_1 = \{0\}$.

Finally, to prove the one-time unforgeability, we just need to remark that knowing signatures of $\boldsymbol{M_1}, \dots, \boldsymbol{M_n} \in \hat{\mathscr{X}}_1$ actually can be seen as knowing a projection key hp' associated to hk when $\hat{\mathscr{L}}_1$ is the subspace generated by $\boldsymbol{\Gamma_1} := \boldsymbol{M_1}, \dots, \boldsymbol{\Gamma_n} := \boldsymbol{M_n}$. Therefore, generating a signature of a message $\boldsymbol{M}$ linearly independent of these messages means generating an NIZK proof for a statement $\boldsymbol{M} \notin \hat{\mathscr{L}}_1$, which has been shown to be hard thanks to the smoothness property of the SPHF and the hard subset membership property of $\mathscr{L}_2$.

One-Round GPAKE. A one-round group password-based authenticated key exchange (GPAKE) is a protocol enabling n users sharing a password pw to establish a common secret key sk in only one round: just by sending one flow. For such protocols, online dictionary attacks, which consist in guessing the password of an honest user and running honestly the protocol with this guessed password, are unavoidable. As a result, the best security that one can hope for is to limit the adversary to at most one password guess per interaction with an honest party. In order to capture this intuition, the formal security model of Abdalla et al. [3], which is recalled in the full version, essentially guarantees that, in a secure GPAKE scheme, no adversary having at most q interactions with honest parties can win with probability higher than q/N, where N is the number of possible passwords. Here, winning means distinguishing a real key (generated by an honest user following the protocol, controlled by the challenger) from a random key sk.

Our construction is a non-trivial extension of the one-round PAKE of Benhamouda et al. in [6], which is an efficient instantiation of the Katz-Vaikuntanathan framework [24]. Basically, a user U_i ($1 \leq i \leq n$) sends an extractable commitment C_i (i.e., an encryption for some public key ek in the common reference string) of his password pw together with a projection key hp_i for the disjunction of $n-1$ languages of valid commitments of pw (words in this disjunction are tuple $\boldsymbol{C}_i = (C_j)_{j \neq i}$ of $n-1$ commitments where at least one of them is a valid commitment of pw). Each partner U_j of this user U_i can compute the hash value H_i of the tuple $\boldsymbol{C}_i$, with hp_i, just by additionally knowing the witness (the random coins) of his commitment C_j onto pw, while U_i uses hk_i. The resulting secret key K is just the XOR of all these hash values (one per hashing key, i.e., one per user): $\mathsf{sk} = H_1 \;\mathsf{xor} \cdots \mathsf{xor}\; H_n$.

At a first glance, one may wonder why our construction relies on a disjunction and not on a conjunction: intuitively, as a user, we would like that every other user commits to the same password as ours. Unfortunately, in this case, nobody would be able to compute the hash value of the expected conjunction, except for the user who generated the hashing key. This is because this computation would require the knowledge of all the witnesses and there is no way for a user to know the witness for a commitment of another user. However, by relying on a disjunction, each user is only required to know the witness for his own commitment.

To understand why this is a secure solution, please note that the challenger (in the security game) can make dummy commitments for the honest players he controls. Then, if no corrupted user (controlled by the adversary) commits to a correct password, the tuple of the $n-1$ other commitments would not be a valid word in the disjunction language (no commitment would be valid) for any of the honest users. Hence, the hash value would appear random to the adversary. The complete proof is a very delicate extension of the proof of the one-round PAKE of Katz and Vaikuntanathan in [24], and may be of independent interest.

Due to recent results by Cheon et al. [9], currently no concrete instantiation of our GPAKE is known for $n \geq 4$ (see Footnote 2 on page 4). For $n = 3$, our scheme only relies on bilinear groups and is practical

2.4 Pseudo-Random Projective Hash Functions and More Efficient Applications

Pseudo-Random Projective Hash Functions. As already explained in Section 1.1, for our (one-time simulation-sound) NIZK and our TSPHF, the second language $\mathscr{L}_2$ is used to provide extra features. Security properties come from its hard subset membership property. However, hard subset membership comes at a cost: the dimension k_2 of $\hat{\mathscr{L}}_2$ has to be at least 1 to be non-trivial, and so the dimension n_2 of $\hat{\mathcal{X}}_2$ is at least 2, otherwise $\hat{\mathscr{L}}_2 = \hat{\mathcal{X}}_2$. This makes the projection key of the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$ of size $k_1 n_2 + n_1 k_2 \geq 2k_1 + n_1$.

Intuitively, what we would like is to be able to have a language $\mathscr{L}_2$ where $n_2 = k_2 = 1$. Such a language would clearly not be hard subset membership, and the smoothness property of SPHF would be completely trivial, since $\hat{\mathcal{X}}_2 \setminus \hat{\mathscr{L}}_2$ would be empty. That is why we introduce the notion of pseudo-randomness which says that the hash value of a word C_2 chosen at random in $\mathcal{X}_2$ (and for implicit languages parameters crs_2 chosen at random), the hash value of C_2 looks random, given only the projection key.

Under DDH in $\mathbb{G}_2$, we can simply choose $\mathsf{crs}_2 = g_2$ a random generator in $\mathbb{G}_2$, $\mathcal{X}_2 = \hat{\mathcal{X}}_2 = \mathscr{L}_2 = \hat{\mathscr{L}}_2 = \mathbb{G}_2$, and θ_2 the identity function. The witness for a word $C_2 \in \mathbb{G}_2$ is just its discrete logarithm in base g_2, and so $\hat{\mathscr{L}}_2$ is seen as generated by the vector $\boldsymbol{\Gamma_1^{(2)}} = (g_2)$. An hashing key hk is just a random scalar $\alpha \in \mathbb{Z}_p$, the associated projection key is $\mathsf{hp} = g_2^\alpha$. Finally the hash value is $H = C_2^\alpha$. It can also be computed using hp if we know the discrete logarithm of C_2. The DDH assumption says that if $g_2, \mathsf{hp} = g_2^\alpha, C_2$ are chosen uniformly at random in $\mathbb{G}_2$, it is hard to distinguish $H = C_2^\alpha$ from a random group element $H \in \mathbb{G}_2$; hence the pseudo-randomness.

Mixed Pseudo-Randomness. In all our applications, we are not really interested in the SPHF on $\mathscr{L}_2$ but in the SPHF on the disjunction $\mathscr{L}$ of $\mathscr{L}_1$ and $\mathscr{L}_2$. Of course, this SPHF would be smooth, but that property is again trivial, since all words (C_1, C_2) are in $\mathscr{L}$. We therefore again need a stronger property called *mixed pseudo-randomness* which roughly says that if hk is a random hashing key, if $C_1 \notin \mathscr{L}_1$ and if C_2 is chosen at random, the hash value of $(C_1, C_2) \in \mathscr{L}$ appears random to any polynomial-time adversary, even given access to the projection key hp.

The proof of this property is quite technical and requires that it is possible to generate parameters of $\mathscr{L}_1$ so that we know the discrete logarithm of the generators $(\boldsymbol{\Gamma}^{(1)}_{i_1})_{i_1}$ of $\hat{\mathscr{L}}_1$. This last property is verified by most languages in which we are interested.

Applications. Using the mixed pseudo-randomness property, we easily get more efficient NIZK and TSPHF, just by replacing $\mathscr{L}_2$ by a language $\mathscr{L}_2$ with a pseudo-random Projective Hash Function. Getting a more efficient one-time simulation-sound NIZK is slightly more complex and is only detailed in the core of the paper. The resulting TSPHF construction actually corresponds to the original construction in [6]. But seeing it as a disjunction of an SPHF for the language we are interested in and of a pseudo-random projective hash function sheds new light on the construction and make it easier to understand, in our opinion.

3 Preliminaries

3.1 Notation

As usual, all the players and the algorithms will be possibly probabilistic and stateful. Namely, adversaries can keep a state st during the different phases, and we denote by $\stackrel{\$}{\leftarrow}$ the outcome of a probabilistic algorithm or the sampling from a uniform distribution. The statement $y \stackrel{\$}{\leftarrow} \mathcal{A}(x; r)$ denotes the operation of running $\mathcal{A}$ with input x and random tape r and storing the result in y. For the sake of clarity, we will sometimes omit the random tape r in $\mathcal{A}(x; r)$.

The qualities of adversaries will be measured by their successes and advantages in certain experiments $\mathsf{Exp}^{\mathsf{sec}}$ or $\mathsf{Exp}^{\mathsf{sec}-b}$ (between the cases $b = 0$ and $b = 1$), denoted $\mathsf{Succ}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})$ respectively, where $\mathfrak{K}$ is the security parameter. Formal definition of all of this and of statistical distance can be found in the full version.

3.2 Definition of SPHF

Let $(\mathscr{L}_{\mathsf{crs}})_{\mathsf{crs}}$ be a family of NP languages indexed by crs with witness relation $\mathcal{R}_{\mathsf{crs}}$, namely $\mathscr{L}_{\mathsf{crs}} = \{x \in \mathcal{X}_{\mathsf{crs}} \mid \exists w,\ \mathcal{R}_{\mathsf{crs}}(x, w) = 1\}$, where $(\mathcal{X}_{\mathsf{crs}})_{\mathsf{crs}}$ is a family set. The value crs is generated by a polynomial-time algorithm $\mathsf{Setup}_{\mathsf{crs}}$ taking as input the unary representation of the security parameter $\mathfrak{K}$, and is usually a common reference string. The description of the underlying group or graded ring is implicit and not part of crs. We suppose that membership in $\mathcal{X}_{\mathsf{crs}}$ and $\mathcal{R}_{\mathsf{crs}}$ can be checked in polynomial time (in $\mathfrak{K}$).

Finally, we suppose that $\mathsf{Setup}_{\mathsf{crs}}$ also outputs a trapdoor $\mathcal{T}_{\mathsf{crs}}$ associated to crs. This trapdoor is empty $\perp$ in most cases, but for some applications (namely NIZK constructions from Section 7), we require that $\mathcal{T}_{\mathsf{crs}}$ contains enough information to decide whether a word $C \in \mathcal{X}$ is in $\mathscr{L}$ or not (or slightly more information). We notice that for most, if not all, languages (we are interested in), it is easy to make $\mathsf{Setup}_{\mathsf{crs}}$ output such a trapdoor, without changing the distribution of crs. In the sequel, crs is often dropped to simplify notation.

An SPHF over $(\mathscr{L}_{\mathsf{crs}})$ is defined by four polynomial-time algorithms:

– $\mathsf{HashKG}(\mathsf{crs})$ generates a hashing key hk;
– $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs})$ derives a projection key hp from hk;
– $\mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, C)$ outputs the hash value from the hashing key, for any crs and for any word $C \in \mathcal{X}$;
– $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{crs}, C, w)$ outputs the hash value from the projection key hp, and the witness w, for a word $C \in \mathscr{L}_{\mathsf{crs}}$ (i.e., $\mathcal{R}_{\mathsf{crs}}(C, w) = 1$).

The set of hash values is called the *range* and is denoted Π. It is often a cyclic group. We always suppose that its size is super-polynomial in the security parameter $\mathfrak{K}$ so that the probability to guess correctly a uniform hash value is negligible.

An SPHF has to satisfy two properties:

– *Perfect correctness*. For any crs and any word $C \in \mathscr{L}_{\mathsf{crs}}$ with witness w (i.e., such that $\mathcal{R}_{\mathsf{crs}}(C, w) = 1$), for any $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs})$ and for $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs})$, $\mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, C) = \mathsf{ProjHash}(\mathsf{hp}, \mathsf{crs}, C, w)$;
– *Perfect smoothness*. The hash value of a word outside the language looks completely random. More precisely, an SPHF is 0-smooth or perfectly smooth if for any crs and any $C \notin \mathscr{L}_{\mathsf{crs}}$, the following two distributions are identical:

$$\Big\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}); H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, C)\Big\}$$
$$\Big\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}); H \xleftarrow{\$} \Pi\Big\} .$$

As shown in the full version, this definition of SPHF actually corresponds to a strong version of KV-SPHF [6] with perfect smoothness[7]. In particular, it is stronger than the definition of SPHF given in [12], where the smoothness is not perfect and is actually defined only for random elements $C \in \mathcal{X} \setminus \mathscr{L}_{\mathsf{crs}}$. This is also slightly stronger than the 1-universal hash proof systems also defined in [12], since the hash value is supposed to look completely random and not just having some minimal entropy.

We restrict ourselves to this very strong form of SPHFs for the sake of simplicity and because most applications we consider require KV-SPHF. However, the construction of disjunctions of SPHFs can still easily be extended to weaker forms of SPHFs.

3.3 Hard Subset Membership Languages

A family of languages $(\mathscr{L}_{\mathsf{crs}} \subseteq \mathcal{X}_{\mathsf{crs}})_{\mathsf{crs}}$ is said to be a hard subset membership family of languages, if is hard to distinguish between a word randomly drawn from inside

[7] The reader familiar with [6] may remark that in our definition, there is no parameter aux in addition to crs. This parameter is indeed useless in the context of KV-SPHFs (contrary to GL-SPHFs), as it can be included in the word C.

$\mathscr{L}_{\mathsf{crs}}$ from a word randomly drawn from outside $\mathscr{L}_{\mathsf{crs}}$ (i.e., from $\mathcal{X}_{\mathsf{crs}} \setminus \mathscr{L}_{\mathsf{crs}}$). This definition implicitly assumes the existence of a distribution over $\mathcal{X}_{\mathsf{crs}}$ and a way to sample efficiently words from $\mathscr{L}_{\mathsf{crs}}$ and from $\mathcal{X}_{\mathsf{crs}} \setminus \mathscr{L}_{\mathsf{crs}}$. This property is formally defined in the full version.

3.4 Bilinear Groups, Graded Rings and Assumptions

All our concrete constructions are based on bilinear groups, which are extensions of cyclic groups. Even though groups should be generated by an appropriate setup algorithm taking the security parameter as input, our definitions below use fixed groups for simplicity.

Cyclic Groups and Bilinear Groups. $(p, \mathbb{G}, g)$ denotes a (multiplicative) cyclic group $\mathbb{G}$ of order p and of generator g. When $(p, \mathbb{G}_1, g_1)$, $(p, \mathbb{G}_2, g_2)$, and $(p, \mathbb{G}_T, g_T)$ are three cyclic groups, $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ or $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is called a *bilinear group* if $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map (called a pairing) efficiently computable and such that $e(g_1, g_2) = g_T$ is a generator of $\mathbb{G}_T$. It is called a *symmetric* bilinear group if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. In this case, we denote it $(p, \mathbb{G}, \mathbb{G}_T, e)$ and we suppose $g = g_1 = g_2$. Otherwise, if $\mathbb{G}_1 \neq \mathbb{G}_2$, it is called an *asymmetric* bilinear group.

Graded Rings. To understand the constructions in the article, it is sufficient to see a graded ring as a way to use ring operations $(+, \bullet)$ over cyclic groups, bilinear groups, or even multilinear groups, as explained at the beginning of Section 2.1. In the sequel, we will often consider two multiplicatively compatible sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$: this basically means that it is possible to compute the product $\bullet$ of any element of $\mathfrak{G}_1$ with any element of $\mathfrak{G}_2$, and the result is in $\mathfrak{G}$. Concretely, as a first approach, it is possible to consider that $\mathfrak{G}$ is a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, and that $\mathfrak{G}_1$ and $\mathfrak{G}_2$ corresponds to $\mathbb{G}_1$ and $\mathbb{G}_2$: if $u_1 \in \mathbb{G}_1$ and $u_2 \in \mathbb{G}_2$, $u_1 \bullet u_2 = e(u_1, u_2)$. General and formal definitions are given in the full version.

Assumptions. The assumption we use the most is the SXDH assumption The SXDH assumption over a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ says the DDH assumption holds both in $(p, \mathbb{G}_1, g_1)$ and $(p, \mathbb{G}_2, g_2)$, where the DDH assumption is defined as follows:

Definition 5 (Decisional Diffie-Hellman (DDH)). *The Decisional Diffie-Hellman assumption says that, in a cyclic group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b, g^c) for unknown random $a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = ab \bmod p$ (a DH tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple).*

We also propose constructions under weaker assumptions than SXDH or DDH, namely κ-Lin, defined as follows:

Definition 6 (κ-Lin). *The κ-Linear assumption says that, in a cyclic group $(p, \mathbb{G}, g)$, when we are given $(g^{a_1}, \dots, g^{a_\kappa}, g^{a_1 b_1}, \dots, g^{a_\kappa b_\kappa}, g^c)$ for unknown $a_1, \dots, a_\kappa, b_1, \dots,$ $b_\kappa \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = b_1 + \cdots + b_\kappa$ (a κ-Lin tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple).*

The 1-Lin assumption is exactly DDH. One advantage of κ-Lin with $\kappa \geq 2$ is that it can hold even in symmetric bilinear groups (where $\mathbb{G}_1 = \mathbb{G}_2$) while DDH or SXDH do not. 2-Lin is also denoted DLin, and κ-Lin often means κ-Lin in $\mathbb{G}_1$ and in $\mathbb{G}_2$. Actually, our constructions can easily be tweaked to support any MDDH assumption [13]. MDDH assumptions generalize κ-Lin assumptions.

4 Smooth Projective Hash Functions for Disjunctions

4.1 Generic Framework and Diverse Vector Spaces

Let us now recall the generic framework for SPHFs. We have already seen the main ideas of this framework in Section 2.1. These ideas were stated in term of generic vector space. Even though using generic vector spaces facilitates the explanation of high level ideas, it is better to use an explicit basis when it comes to details. As already explained in Section 2.1 on page 7, compared to [6], all vectors and matrices are transposed.

Let $\mathfrak{G}$ be a graded ring. We now set $\hat{\mathscr{X}} = \mathfrak{G}^n$, so that any vector $\hat{\boldsymbol{C}} \in \hat{\mathscr{X}}$ is a n-dimensional *column* vector. We denote by $(\boldsymbol{e_i})_{i=1}^n$ the canonical basis of $\hat{\mathscr{X}}$. The dual space of $\hat{\mathscr{X}}$ is isomorphic[8] to $\mathbb{Z}_p^{1\times n}$, and the hashing key $\alpha \in \hat{\mathscr{X}}^*$ corresponds to the *row* vector $\boldsymbol{\alpha} = (\alpha_i)_{i=1}^n$, with $\alpha_i = \alpha(\boldsymbol{e_i})$. We denote by Γ the matrix with columns $(\boldsymbol{\Gamma_i})_{i=1}^k$, where the family $(\boldsymbol{\Gamma_i})$ generates the subspace $\hat{\mathscr{L}}$ of $\hat{\mathscr{X}}$. Finally, we assume that for each coordinate of the vector $\theta(C) \in \mathfrak{G}^n$, the group in which it is (called the index of the coordinate, in the formal description of graded rings in the full version) is independent of C.

We suppose that, a word $C \in \mathscr{X}$ is in $\mathscr{L}$ if and only if there exists $\boldsymbol{\lambda} \in \mathfrak{G}^k$ such that $\hat{\boldsymbol{C}} := \theta(C) = \Gamma \bullet \boldsymbol{\lambda}$. We also assume the latter equality holds if and only if it would hold by only looking at the discrete logarithms (and not at the groups or indexes of entries or coordinates)[9]. In addition, we suppose that $\boldsymbol{\lambda}$ can be computed easily from any witness w for C; and in the sequel we often simply consider that $w = \boldsymbol{\lambda}$. By analogy with diverse groups [12], as explained in Section 2.1, we say that a tuple $\mathcal{V} = (\mathscr{X}, \mathscr{L}, \mathcal{R}, \mathfrak{G}, n, k, \Gamma, \theta)$ satisfying the above properties is a *diverse vector space*.

A summary of diverse vector spaces and the construction of SPHF over them can be found in Fig. 1. It is straightforward to see (and this is proven in [6]) that any SPHF defined by a discrete vector space $\mathcal{V}$ as in Fig. 1 is correct and smooth.

4.2 Disjunctions of SPHFs

As explained in Section 2.1, an SPHF for the disjunction of two languages $\mathscr{L}_1$ and $\mathscr{L}_2$ roughly consists in doing the tensor product of their related vector spaces $\hat{\mathscr{X}}_1$ and $\hat{\mathscr{X}}_2$. However, our vector spaces are not classical vector spaces, since they are over graded rings. In particular, multiplication of scalars is not always possible, and so tensor product may not be always possible either. That is why, we first need to introduce the notion of tensor product of vector spaces over graded rings, before giving the detailed construction of disjunctions of SPHFs.

Tensor Product of Vector Spaces over Graded Rings. Let us very briefly recall notations for tensor product and adapt them to vector spaces over graded rings. Let $\mathfrak{G}_1$ and $\mathfrak{G}_2$

[8] Here we consider $\hat{\mathscr{X}}$ as $\mathbb{Z}_p^n$, for the sake of simplicity.

[9] Formal requirements can be found in the full version.

Diverse Vector Space $\mathcal{V}$ = tuple $(\mathcal{X}, \mathscr{L}, \mathcal{R}, \mathfrak{G}, n, k, \Gamma, \theta)$ where

- n, k are two positive integers;
- $\mathscr{L} \subset \mathcal{X}$ is an NP-language, defined by a witness relation $\mathcal{R}$ (and implicitely indexed by crs):
$$\mathscr{L}_{\mathsf{crs}} = \{x \in \mathcal{X}_{\mathsf{crs}} \mid \exists \boldsymbol{\lambda} \in \mathfrak{G}^k,\ \mathcal{R}_{\mathsf{crs}}(x, \boldsymbol{\lambda}) = 1\}$$
- $\mathfrak{G}$ is a graded ring;
- θ is a function from $\mathscr{L}$ to $\mathfrak{G}^n$; *notation:* $\hat{\boldsymbol{C}} := \theta(C)$;
- Γ is a matrix in $\mathfrak{G}^{n \times k}$;

such that for $C \in \mathcal{X}$ and $\boldsymbol{\lambda} \in \mathfrak{G}^k$:

$$\mathcal{R}(x, \boldsymbol{\lambda}) = 1 \qquad \Longleftrightarrow \qquad \hat{\boldsymbol{C}} := \theta(C) = \Gamma \bullet \boldsymbol{\lambda},$$

plus some additional technical assumptions on groups or indexes of coefficients of $\theta(C)$ and Γ (see text).
Notation: $\hat{\mathscr{L}}$ is the vector space generated by the columns of Γ;

SPHF for $\mathcal{V} = (\mathcal{X}, \mathscr{L}, \mathcal{R}, \mathfrak{G}, n, k, \Gamma, \theta)$:

$\mathsf{HashKG}(\mathsf{crs})$	outputs a random row vector $\mathsf{hk} := \boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{1 \times n}$
$\mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs})$	outputs $\mathsf{hp} := \boldsymbol{\gamma} = \boldsymbol{\alpha} \bullet \Gamma \in \mathfrak{G}^{1 \times k}$
$\mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, C)$	outputs $H := \boldsymbol{\alpha} \bullet \hat{\boldsymbol{C}} \in \mathfrak{G}$
$\mathsf{ProjHash}(\mathsf{hp}, \mathsf{crs}, C, \boldsymbol{\lambda})$	outputs $H' := \boldsymbol{\gamma} \bullet \boldsymbol{\lambda} \in \mathfrak{G}$

Fig. 1. Diverse Vector Space and Smooth Projective Hash Function (SPHF)

be two multiplicatively compatible sub-graded rings of $\mathfrak{G}$. Let V_1 be a n_1-dimensional vector space over $\mathfrak{G}_1$ and V_2 be a n_2-dimensional vector space over $\mathfrak{G}_2$. Let $(\boldsymbol{e_{1,i}})_{i=1}^{n_1}$ and $(\boldsymbol{e_{2,i}})_{i=1}^{n_2}$ be bases of V_1 and V_2 respectively. Then the *tensor product* V of V_1 and V_2, denoted $V = V_1 \otimes V_2$ is the $n_1 n_2$-dimensional vector space over $\mathfrak{G}$ generated by the free family $(\boldsymbol{e_{1,i}} \otimes \boldsymbol{e_{2,j}})_{\substack{i=1,\dots,n_1 \\ j=1,\dots,n_2}}$. The operator $\otimes$ is bilinear, and if $\boldsymbol{u} = \sum_{i=1}^{n_1} u_i \bullet \boldsymbol{e_{1,i}}$ and $\boldsymbol{v} = \sum_{j=1}^{n_2} v_j \bullet \boldsymbol{e_{2,j}}$, then:

$$\boldsymbol{u} \otimes \boldsymbol{v} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (u_i \bullet v_j) \bullet (e_{1,i} \otimes e_{2,j}).$$

More generally, we can define the tensor product of two matrices $M \in \mathfrak{G}_1^{k \times m}$ and $M' \in \mathfrak{G}_2^{k' \times m'}$, $T = M \otimes M' \in \mathfrak{G}^{kk' \times mm'}$ by

$$T_{(i-1)k'+i',(j-1)m'+j'} = M_{i,j} \bullet M'_{i',j'} \qquad \text{for} \begin{cases} i = 1, \dots, k,\ i' = 1, \dots, k', \\ j = 1, \dots, m,\ j' = 1, \dots, m'. \end{cases}$$

And if $M \in \mathfrak{G}_1^{k \times m}$, $M' \in \mathfrak{G}_2^{k' \times m'}$, $N \in \mathfrak{G}_1^{m \times n}$ and $N' \in \mathfrak{G}_2^{m' \times n'}$, and if $M \bullet N$ and $M' \bullet N'$ are well-defined (i.e., index of coefficients are "coherent"), then we have

$$(M \otimes M') \bullet (N \otimes N') = (M \bullet N) \otimes (M' \bullet N').$$

Finally, this definition can be extended to more than 2 vector spaces.

Disjunctions of SPHFs. In Fig. 2, we show the construction of the disjunction of two diverse vector spaces, over two multiplicatively sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. In applications, we will often have $\mathfrak{G}_1 = \mathbb{G}_1$ and $\mathfrak{G}_2 = \mathbb{G}_2$ where $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ is a bilinear group.

Diverse vector space $\mathcal{V} = (\mathcal{X}, \mathscr{L}, \mathcal{R}, \mathfrak{G}, n, k, \Gamma, \theta)$ **disjunction** of diverse vector spaces $\mathcal{V}_1 = (\mathcal{X}_1, \mathscr{L}_1, \mathcal{R}_1, \mathfrak{G}_1, n_1, k_1, \Gamma_1, \theta_1)$ and $\mathcal{V}_2 = (\mathcal{X}_2, \mathscr{L}_2, \mathcal{R}_2, \mathfrak{G}_2, n_2, k_2, \Gamma_2, \theta_2)$:

- $\mathfrak{G}_1$ and $\mathfrak{G}_2$ are two multiplicatively compatible sub-graded rings of $\mathfrak{G}$;
- $n = n_1 n_2$ $\qquad$ $k = k_1 n_2 + n_1 k_2$;
- $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ $\qquad$ $\mathscr{L} = (\mathscr{L}_1 \times \mathcal{X}_2) \cup (\mathcal{X}_1 \times \mathscr{L}_2)$
- $\Gamma = \begin{pmatrix} \Gamma^{(1)} \otimes \mathsf{Id}_{n_2} & \mathsf{Id}_{n_1} \otimes \Gamma^{(2)} \end{pmatrix}$ $\qquad$ $\theta((C_1, C_2)) = \hat{C}1 \otimes \hat{C}2$;
- Witnesses $\boldsymbol{\lambda}$ for $C = (C_1, C_2) \in \mathscr{L}$ (i.e., vectors $\boldsymbol{\lambda} \in \mathfrak{G}^k$ such that $\mathcal{R}(C, \boldsymbol{\lambda}) = 1$) are:

$$\boldsymbol{\lambda} = \begin{cases} \begin{pmatrix} \boldsymbol{\lambda_1} \otimes \hat{\boldsymbol{C}}_2 \\ \boldsymbol{0} \in \mathbb{Z}_p^{n_1 k_2} \end{pmatrix} & \text{when } \mathcal{R}_1(C_1, \boldsymbol{\lambda_1}) = 1 \\ \begin{pmatrix} \boldsymbol{0} \in \mathbb{Z}_p^{k_1 n_2} \\ \hat{\boldsymbol{C}}_1 \otimes \boldsymbol{\lambda_2} \end{pmatrix} & \text{when } \mathcal{R}_2(C_2, \boldsymbol{\lambda_2}) = 1 \end{cases}$$

for any $C = (C_1, C_2) \in \mathcal{X}$ and any $\boldsymbol{\lambda} \in \mathfrak{G}^n$.

Notation: Due to the form of witnesses, we split $\boldsymbol{\gamma}$ in two parts: $\boldsymbol{\gamma^{(1)}}$ corresponds to the first $k_1 n_2$ columns of $\boldsymbol{\gamma}$, while $\boldsymbol{\gamma^{(2)}}$ corresponds to the last $k_2 n_1$ columns of $\boldsymbol{\gamma}$.

Fig. 2. Disjunction of Diverse Vector Spaces

Let us explain this construction. First, the rows of Γ generate the following subspace of $\hat{\mathcal{X}} = \mathfrak{G}^{1 \times n} = \hat{\mathcal{X}}_1 \otimes \hat{\mathcal{X}}_2$:

$$\hat{\mathscr{L}} = \langle (\hat{\mathscr{L}}_1 \otimes \hat{\mathcal{X}}_2) \cup (\hat{\mathcal{X}}_1 \otimes \hat{\mathscr{L}}_2) \rangle,$$

where $\hat{\mathcal{X}}_1 = \mathfrak{G}_1^{n_1}$, $\hat{\mathcal{X}}_2 = \mathfrak{G}_2^{n_2}$, $\hat{\mathscr{L}}_1$ is the subspace of $\hat{\mathcal{X}}_1$ generated by the rows of $\Gamma^{(1)}$ and $\hat{\mathscr{L}}_2$ is the subspace of $\hat{\mathcal{X}}_2$ generated by the rows of $\Gamma^{(2)}$. So this construction corresponds exactly to the one sketched in the Section 2.1.

Then, we need to prove that $\mathcal{V}$ is really a diverse vector space, namely that $C \in \mathscr{L}$ if and only if $\theta(C) \in \hat{\mathscr{L}}$. Clearly, if $C = (C_1, C_2) \in \mathscr{L}$, then $\hat{\boldsymbol{C}}_1 \in \hat{\mathscr{L}}_1$ or $\hat{\boldsymbol{C}}_2 \in \hat{\mathscr{L}}_2$ and so $\hat{\boldsymbol{C}} = \hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2 \in \hat{\mathscr{L}}$. Now, let us prove the converse. Let $C = (C_1, C_2) \notin \mathscr{L}$. So, $\hat{\boldsymbol{C}}_1 \notin \hat{\mathscr{L}}_1$ and $\hat{\boldsymbol{C}}_2 \notin \hat{\mathscr{L}}_2$. Let H_1 and H_2 be supplementary vector spaces of $\hat{\mathscr{L}}_1$ and $\hat{\mathscr{L}}_2$ (in $\hat{\mathcal{X}}_1$ and $\hat{\mathcal{X}}_2$, respectively). Then $\hat{\mathcal{X}}_1$ is the direct sum of $\hat{\mathscr{L}}_1$ and H_1, while $\hat{\mathcal{X}}_2$ is the direct sum of $\hat{\mathscr{L}}_2$ and H_2. Therefore, $\hat{\mathscr{L}}_1 \otimes \hat{\mathcal{X}}_2$ is the direct sum of $\hat{\mathscr{L}}_1 \otimes \hat{\mathscr{L}}_2$ and $\hat{\mathscr{L}}_1 \otimes H_2$, while $\hat{\mathcal{X}}_1 \otimes \hat{\mathscr{L}}_2$ is the direct sum of $\hat{\mathscr{L}}_1 \otimes \hat{\mathscr{L}}_2$ and $H_1 \otimes \hat{\mathscr{L}}_2$. So finally, $\hat{\mathscr{L}}$ is the direct sum of $\hat{\mathscr{L}}_1 \otimes \hat{\mathscr{L}}_2$, $\hat{\mathscr{L}}_1 \otimes H_2$ and $H_1 \otimes \hat{\mathscr{L}}_2$; and $H_1 \otimes H_2$ is a supplementary of $\hat{\mathscr{L}}$. Since $0 \neq \hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2 \in H_1 \otimes H_2$, $\theta(C) = \hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2 \notin \hat{\mathscr{L}}$.

Besides showing the correctness of the construction, this proof helps to better understand the structure of $\hat{\mathscr{L}}$. In particular, it shows that $\hat{\mathscr{L}}$ has dimension $l_1 l_2 + (n_1 - l_1) l_2 + l_1 (n_2 - l_2) = l_1 n_2 + n_1 l_2 - l_1 l_2$, if $\hat{\mathscr{L}}_1$ has dimension l_1 and $\hat{\mathscr{L}}_2$ has dimension l_2. If the rows of $\Gamma^{(1)}$ and $\Gamma^{(2)}$ are linearly independent, $l_1 = k_1$ and $l_2 = k_2$, $\hat{\mathscr{L}}$ has dimension $k_1 n_2 + n_1 k_2 - k_1 k_2$, which is less than $k_1 n_2 + n_1 k_2$, the number of rows of Γ. Therefore the rows of Γ are never linearly independent. Actually, this last result can directly be proven by remarking that if $\hat{\boldsymbol{C}}_1 \in \hat{\mathscr{L}}_1$ and $\hat{\boldsymbol{C}}_2 \in \hat{\mathscr{L}}_2$, then $\hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2 \in (\hat{\mathscr{L}}_1 \otimes \hat{\mathcal{X}}_2) \cap (\hat{\mathcal{X}}_1 \otimes \hat{\mathscr{L}}_2)$. For the sake of completeness, detailed and concrete equations are detailed in the full version.

5 One-Time Simulation-Sound NIZK from Disjunctions of SPHFs

In this section, we present our construction of NIZK and one-time simulation-sound NIZK from disjunctions of SPHFs. The latter requires the use of a new notion: 2-smooth projective hash functions. We suppose the reader is familiar with NIZK and one-time simulation-sound NIZK. Formal definitions can be found in the full version.

5.1 NIZK from Disjunctions of SPHFs

Construction. In Fig. 3, we show how to construct a NIZK for any family of languages $\mathscr{L}_1$ such that there exist two diverse vector spaces $\mathcal{V}_1 = (\mathcal{X}_1, \mathscr{L}_1, \mathcal{R}_1, \mathfrak{G}_1, n_1, k_1, \Gamma^{(1)}, \theta_1)$ and $\mathcal{V}_2 = (\mathcal{X}_2, \mathscr{L}_2, \mathcal{R}_2, \mathfrak{G}_2, n_2, k_2, \Gamma^{(2)}, \theta_2)$ over two multiplicatively-compatible sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$, such that the second diverse vector space corresponds to a hard subset membership language. In particular, this construction works for any diverse vector space $\mathcal{V}_1$ where $\mathfrak{G}_1 = \mathbb{G}_1$ is a cyclic group of some bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where SXDH holds, by using as $\mathcal{V}_2$ the discrete vector space for DDH over $\mathbb{G}_2$ (Example 1).

The proof π of a word C_1 can just be seen as the hash values of rows[10] of $\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{n_2}$. Let us now show that our NIZK is complete, zero-knowledge and sound.

Completeness. If the proof $\boldsymbol{\pi}$ has been generated correctly, the left hand side of the verification equation (Eq. (1)) is equal to

$$\begin{aligned}
\boldsymbol{\gamma^{(1)}} \bullet (\boldsymbol{\lambda_1} \otimes \mathsf{Id}_{n_2}) \bullet \Gamma^{(2)} &= (\boldsymbol{\alpha} \bullet (\Gamma^{(1)} \otimes \mathsf{Id}_{n_2})) \bullet (\boldsymbol{\lambda_1} \otimes \mathsf{Id}_{n_2}) \bullet (\mathsf{Id}_1 \otimes \Gamma^{(2)}) \\
&= \boldsymbol{\alpha} \bullet (\Gamma^{(1)} \otimes \mathsf{Id}_{n_2}) \bullet ((\boldsymbol{\lambda_1} \bullet \mathsf{Id}_1) \otimes (\mathsf{Id}_{n_2} \bullet \Gamma^{(2)})) \\
&= \boldsymbol{\alpha} \bullet (\Gamma^{(1)} \otimes \mathsf{Id}_{n_2}) \bullet (\boldsymbol{\lambda_1} \otimes \Gamma^{(2)}) \\
&= \boldsymbol{\alpha} \bullet ((\Gamma^{(1)} \bullet \boldsymbol{\lambda_1}) \otimes (\mathsf{Id}_{n_2} \bullet \Gamma^{(2)})),
\end{aligned}$$

while the right hand side is always equal to:

$$\boldsymbol{\gamma^{(2)}} \bullet (\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{k_2}) = \boldsymbol{\alpha} \bullet (\mathsf{Id}_{n_1} \otimes \Gamma^{(2)}) \bullet (\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{k_2}) = \boldsymbol{\alpha} \bullet ((\mathsf{Id}_{n_1} \bullet \hat{\boldsymbol{C}}_1) \otimes (\Gamma^{(2)} \bullet \mathsf{Id}_{k_2})),$$

[10] This is not quite accurate, since rows of $\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{n_1}$ are not words in $\mathcal{X}$ but in $\hat{\mathcal{X}}$. But to give intuition, we will often make this abuse of notation.

NIZK for $\mathcal{V}_1 = (\mathcal{X}_1, \mathscr{L}_1, \mathcal{R}_1, \mathfrak{G}_1, n_1, k_1, \Gamma_1, \theta_1)$, using $\mathcal{V}_2 = (\mathcal{X}_2, \mathscr{L}_2, \mathcal{R}_2, \mathfrak{G}_2, n_2, k_2, \Gamma_2, \theta_2)$, with $\mathscr{L}_2$ a hard subset membership language:

- $\mathcal{V} = (\mathcal{X}, \mathscr{L}, \mathcal{R}, \mathfrak{G}, n, k)$ disjunction of $\mathcal{V}_1$ and $\mathcal{V}_2$;
- Setup: computes $\mathsf{hk} = \boldsymbol{\alpha} \overset{\$}{\leftarrow} \mathbb{Z}_p^{1\times n}$, $\mathsf{hp} = \boldsymbol{\gamma} = \Gamma \bullet \boldsymbol{\alpha}$, and outputs trapdoor $\mathcal{T} := \mathsf{hk}$, and CRS $\sigma := (\mathsf{crs}_2, \mathsf{hp})$;
- Proof $\boldsymbol{\pi}$ of $C_1 \in \mathscr{L}_1$ with witness $\boldsymbol{\lambda_1} \in \mathfrak{G}^{k_1}$:

$$\boldsymbol{\pi} := \boldsymbol{\gamma^{(1)}} \bullet (\boldsymbol{\lambda_1} \otimes \mathsf{Id}_{n_2}) \in \mathfrak{G}_1^{1\times n_2};$$

- Verification of proof $\boldsymbol{\pi}$ for C_1:

$$\boldsymbol{\pi} \bullet \Gamma^{(2)} \overset{?}{=} \boldsymbol{\gamma^{(2)}} \bullet (\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{k_2}), \tag{1}$$

- Simulation of proof $\boldsymbol{\pi}$ for C_1 knowing $\mathcal{T} = \mathsf{hk}$:

$$\boldsymbol{\pi} := \boldsymbol{\alpha} \bullet (\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{n_2}).$$

Fig. 3. NIZK from Disjunctions of Diverse Spaces

which is the same as the left hand side, since $\Gamma^{(1)} \bullet \boldsymbol{\lambda_1} = \mathsf{Id}_{n_1} \bullet \hat{\boldsymbol{C}}_1$ and $\mathsf{Id}_{n_2} \bullet \Gamma^{(2)} = \Gamma^{(2)} \bullet \mathsf{Id}_{k_2}$. Hence the *completeness*. Another way to see it, is that the row i_2 of the right hand side is the hash value of "$(\hat{\boldsymbol{C}}_1, \Gamma^{(2)} \bullet \boldsymbol{e_{2,i_2}})$" computed using the witness $\boldsymbol{\lambda_2} = \boldsymbol{e_{2,i_2}}$, while the row i_2 of the left hand side is this hash value computed using the witness $\boldsymbol{\lambda_1}$.

Zero-Knowledge. The (perfect) unbounded *zero-knowledge* property comes from the fact that the normal proof $\boldsymbol{\pi}$ for $C_1 \in \mathscr{L}_1$ with witness $\boldsymbol{\lambda_1}$ is:

$$\boldsymbol{\gamma^{(1)}} \bullet (\boldsymbol{\lambda_1} \otimes \mathsf{Id}_{n_2}) = \boldsymbol{\alpha} \bullet (\Gamma^{(1)} \otimes \mathsf{Id}_{n_2}) \bullet (\boldsymbol{\lambda_1} \otimes \mathsf{Id}_{n_2}) = \boldsymbol{\alpha} \bullet ((\Gamma^{(1)} \bullet \boldsymbol{\lambda_1}) \otimes (\mathsf{Id}_{n_2} \bullet \mathsf{Id}_{n_2})),$$

which is equal to the simulated proof for C_1, as $\hat{\boldsymbol{C}}_1 = \Gamma^{(1)} \bullet \boldsymbol{\lambda_1}$ and $\mathsf{Id}_{n_2} \bullet \mathsf{Id}_{n_2} = \mathsf{Id}_{n_2}$.

Soundness. It remains to prove the soundness property, under the hard subset membership of $\mathscr{L}_2$. We just need to show that if the adversary is able to generate a valid proof $\boldsymbol{\pi}$ for a word $C_1 \notin \mathscr{L}_1$, then we can use $\boldsymbol{\pi}$ to check if a word C_2 is in $\mathscr{L}_2$ or not. More precisely, let $C_2 \in \mathcal{X}_2$, let H be the hash value of (C_1, C_2) computed using hk, and let us define $H' := \boldsymbol{\pi} \bullet \hat{\boldsymbol{C}}_2$.

On the one hand, if $C_2 \in \mathscr{L}_2$, there exists a witness $\boldsymbol{\lambda_2}$ such that $\hat{\boldsymbol{C}}_2 = \Gamma^{(2)} \bullet \boldsymbol{\lambda_2}$ and so, thanks to (1):

$$H' = \boldsymbol{\pi} \bullet \Gamma^{(2)} \bullet \boldsymbol{\lambda_2} = \boldsymbol{\gamma^{(2)}} \bullet (\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{k_2}) \bullet \boldsymbol{\lambda_2} = \boldsymbol{\gamma^{(2)}} \bullet (\hat{\boldsymbol{C}}_1 \otimes \boldsymbol{\lambda_2}) = H,$$

the last equality coming from the correctness of the SPHF and the fact the last-but-one expression is just the hash value of (C_1, C_2) computed using ProjHash and witness $\boldsymbol{\lambda_2}$.

On the other hand, if $C_2 \notin \mathscr{L}_2$, then $(C_1, C_2) \notin \mathscr{L}$. So H looks completely random by smoothness and the probability that $H' = H$ is at most $1/|\Pi|$.

Toward One-Time Simulation Soundness. The previous proof does not work anymore if the adversary is allowed to get even one single simulated proof of a word $C_1 \notin \mathscr{L}_1$.

Indeed, in this case, the smoothness does not hold anymore, in the above proof of soundness. That is why we need a stronger form of smoothness for SPHF, called 2-smoothness.

5.2 2-Smooth Projective Hash Functions

Definition. In order to define the notion of 2-smoothness, let us first introduce the notion of tag-SPHF. A *tag-SPHF* is similar to an SPHF except that Hash and $\mathsf{ProjHash}$ now take a new input, called a tag $\mathsf{tag} \in \mathsf{Tags}$. Similarly a *tag diverse vector space* is a diverse vector space where the function θ also takes as input a tag $\mathsf{tag} \in \mathbb{Z}_p$. The vector $\boldsymbol{\lambda}$ is now allowed to depend on tag, but the matrix Γ is independent of tag.

A 2-smooth SPHF is a tag-SPHF for which the hash value of a word $C \in \mathcal{X}$ for a tag tag looks random even if we have access to the hash value of another word $C' \in \mathcal{X}$ for a different tag $\mathsf{tag}' \neq \mathsf{tag}$. Formally, a tag-$\mathsf{SPHF}$ is perfectly 2-smooth, if for any crs, any $C' \in \mathcal{X}$, any distinct tags $\mathsf{tag}, \mathsf{tag}'$, and any $C \notin \mathscr{L}_{\mathsf{crs}}$, the following two distributions are identical:

$$\left\{ (\mathsf{hp}, H', H) \;\middle|\; \begin{array}{ll} \mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs}); & \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}); \\ H' \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, (C', \mathsf{tag}')); & H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, (C, \mathsf{tag})) \end{array} \right\}$$

$$\left\{ (\mathsf{hp}, H', H) \;\middle|\; \begin{array}{ll} \mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs}); & \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}); \\ H' \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, (C', \mathsf{tag}')); & H \stackrel{\$}{\leftarrow} \Pi \end{array} \right\}.$$

A weaker (statistical instead of perfect) definition is proposed in the full version. The 2-smoothness property is similar to the 2-universality property in [12]. There are however two minor differences, the first being the existence of an explicit tag, and the second being that the hash value of a word outside the language is supposed to be uniformly random instead of just having some entropy. This slightly simplifies its usage in our constructions, in our opinion.

Canonical Construction from Diverse Vector Spaces. Let $\mathcal{V} = (\mathcal{X}, \mathscr{L}, \mathcal{R}, \mathfrak{G}, n, k, \Gamma, \theta)$ be a diverse vector space. If we set $\tilde{n} = 2n$, $\tilde{k} = 2k$, and:

$$\tilde{\Gamma} = \begin{pmatrix} \Gamma & 0 \\ 0 & \Gamma \end{pmatrix} \qquad \tilde{\boldsymbol{\lambda}} = \begin{pmatrix} \boldsymbol{\lambda} \\ \mathsf{tag} \bullet \boldsymbol{\lambda} \end{pmatrix} \qquad \tilde{\theta}(C, \mathsf{tag}) = \begin{pmatrix} \hat{\boldsymbol{C}} \\ \mathsf{tag} \bullet \hat{\boldsymbol{C}} \end{pmatrix},$$

where $\tilde{\boldsymbol{\lambda}}$ is the witness for a word $C \in \mathscr{L}$ and a tag tag, then $\tilde{\mathcal{V}} = (\mathcal{X}, \mathscr{L}, \mathcal{R}, \mathfrak{G}, \tilde{n}, \tilde{k}, \tilde{\Gamma}, \tilde{\theta})$ is a 2-smooth diverse vector space. It is clear that $C \in \mathscr{L}$ if and only if $\tilde{\hat{\boldsymbol{C}}} = \tilde{\theta}(C, \mathsf{tag})$ is a linear combination of rows of Γ.

To prove the 2-smoothness property, let $C' \in \mathcal{X}$ and $C \in \mathcal{X} \setminus \mathscr{L}$, and let tag' and tag be two distinct tags. We have

$$\tilde{\hat{\boldsymbol{C}}}' = \begin{pmatrix} \hat{\boldsymbol{C}}' \\ \mathsf{tag}' \bullet \hat{\boldsymbol{C}}' \end{pmatrix} \qquad \text{and} \qquad \tilde{\hat{\boldsymbol{C}}} = \begin{pmatrix} \hat{\boldsymbol{C}} \\ \mathsf{tag} \bullet \hat{\boldsymbol{C}} \end{pmatrix}.$$

We just need to prove that $\tilde{\hat{\boldsymbol{C}}}$ is not in the subspace generated by the rows of Γ and $\tilde{\hat{\boldsymbol{C}}}'$, or in other words that it is not in $\hat{\mathscr{L}}' = \langle \hat{\mathscr{L}} \cup \{\tilde{\hat{\boldsymbol{C}}}'\} \rangle$. Indeed, in that case, H' could just be seen as a part of the projection key for the language $\hat{\mathscr{L}}'$, and by smoothness, we get that H looks uniformly random.

So it remains to prove that linear independence of $\tilde{\hat{\boldsymbol{C}}}$. By contradiction, let us suppose there exists $\tilde{\boldsymbol{\lambda}} \in \mathbb{Z}_p^{2k}$ and μ such that:

$$\tilde{\hat{\boldsymbol{C}}} = \begin{pmatrix} \hat{\boldsymbol{C}} \\ \mathsf{tag} \bullet \hat{\boldsymbol{C}} \end{pmatrix} = \tilde{\Gamma} \bullet \tilde{\boldsymbol{\lambda}} + \tilde{\hat{\boldsymbol{C}}}' \bullet \mu = \begin{pmatrix} \Gamma & 0 \\ 0 & \Gamma \end{pmatrix} \bullet \tilde{\boldsymbol{\lambda}} + \begin{pmatrix} \hat{\boldsymbol{C}}' \\ \mathsf{tag}' \bullet \hat{\boldsymbol{C}}' \end{pmatrix} \bullet \mu.$$

Therefore $\tilde{\hat{\boldsymbol{C}}} + \mu \bullet \tilde{\hat{\boldsymbol{C}}}'$ and $\mathsf{tag} \bullet \tilde{\hat{\boldsymbol{C}}} + \mathsf{tag}' \bullet \mu \bullet \tilde{\hat{\boldsymbol{C}}}'$ are both linear combination of rows of Γ, and so is

$$\mathsf{tag}' \bullet (\tilde{\hat{\boldsymbol{C}}} + \mu \bullet \tilde{\hat{\boldsymbol{C}}}') \;+\; (\mathsf{tag} \bullet \tilde{\hat{\boldsymbol{C}}} + \mathsf{tag}' \bullet \mu \bullet \tilde{\hat{\boldsymbol{C}}}') \;=\; (\mathsf{tag}' - \mathsf{tag}) \bullet \tilde{\hat{\boldsymbol{C}}}.$$

As $\mathsf{tag}' - \mathsf{tag} \neq 0$, this implies that $\tilde{\hat{\boldsymbol{C}}}$ is also a linear combination of rows of Γ, hence $C \in \mathscr{L}$, which is not the case.

5.3 One-Time Simulation-Sound Zero-Knowledge Arguments from SPHF

Let us now replace the first diverse vector space by its canonical 2-smooth version in the NIZK construction of Section 5.1. The resulting construction is a one-time simulation-sound NIZK, if $\hat{\boldsymbol{C}}_1$ is computed as $\theta_1(C_1, \mathsf{tag})$ where tag is the hash value of (C_1, ℓ) under some collision-resistant hash function $\mathcal{H}$: $\mathsf{tag} = \mathcal{H}((C_1, \ell))$.

Completeness and perfect zero-knowledge can be proven the same way. It remains to prove the one-time simulation soundness. The proof is similar to the one in Section 5.1, except for the final step: proving that the hash value H of (C_1, C_2) with tag $\mathsf{tag} = \mathcal{H}((C_1, \ell))$ looks random even if the adversary sees a simulated NIZK π' for a word $C_1' \in \mathcal{X}_1$ and label ℓ'.

We first remark that the tag tag' can be supposed distinct from the tag tag for the NIZK π created by the adversary, thanks to the collision-resistance of $\mathcal{H}$. We recall that π' is the hash values of the rows of $\hat{\boldsymbol{C}}_1' \otimes \mathsf{Id}_{n_2}$. So to prove that the hash value of (C_1, C_2) with tag tag looks random even with access to π', we just need to remark that $\hat{\boldsymbol{C}}_1 \otimes \hat{\boldsymbol{C}}_2$ is linearly independent of rows of Γ and $\hat{\boldsymbol{C}}_1' \otimes \mathsf{Id}_{n_2}$. The proof is similar to the proof of 2-smoothness.

Remark 7. It would be easy to extend this construction to handle N-time simulation-sound NIZK, for any constant N. The NIZK CRS σ size would just be N times larger compared to the NIZK construction of Section 5.1, and the proof size would remain constant.

5.4 Concrete Instantiation

If $\mathcal{V}_1$ is a diverse vector space over $\mathbb{G}_1$ and $\mathcal{V}_2$ is the diverse vector space for DDH in $\mathbb{G}_2$, where $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ is a bilinear group where DDH is hard in $\mathbb{G}_2$, then we get a NIZK and a one-time simulation sound NIZK whose proof is composed of only $n_2 = 2$ group elements in $\mathbb{G}_1$.

More generally, we can use as $\mathcal{V}_2$, any diverse vector space from any MDDH assumption [13]. Under κ-Lin, we get a proof consisting of only $n_2 = \kappa + 1$ group elements. Details can be found in the full version.

Languages handled are exactly languages for which there exists such a diverse vector space $\mathcal{V}_1$ over $\mathbb{G}_1$. That corresponds to languages handled by Jutla and Roy NIZK [20], which they call linear subspaces (assuming θ is the identity function), if we forget the fact that in [20], it is supposed that crs can be generated in such a way that discrete logarithms of Γ is known (that is what they call *witness-samplable* languages). That encompasses DDH, κ-Lin, and languages of ElGamal, Cramer-Shoup or similar ciphertexts whose plaintexts verify some linear system of equations, as already shown in [6]. Concrete comparison with previous work can be found in Section 7.3.

5.5 Application: Threshold Cramer-Shoup-like Encryption Scheme

The Cramer-Shoup public-key encryption scheme [11] is one of the most efficient $\mathsf{IND\text{-}CCA}$ encryption schemes with a proof of security in the standard model. We remark here that, if we replace the last part of a Cramer-Shoup ciphertext (the 2-universal projective hash proof) by a one-time simulation-sound NIZK on the DDH language, we can obtain an $\mathsf{IND\text{-}CCA}$ scheme supporting efficient threshold decryption. Intuitively, this comes from the fact that the resulting scheme becomes "publicly verifiable", in the sense that, after verifying the NIZK (which is publicly verifiable), one can obtain the underlying message via "simple" algebraic operations which can easily be "distributed".

Previous one-time simulation-sound NIZK were quite inefficient and the resulting scheme would have been very inefficient compared to direct constructions of threshold $\mathsf{IND\text{-}CCA}$ encryption schemes. However, in our case, our new one-time simulation-sound NIZK based on disjunctions of SPHF only adds one group element to the ciphertext (compared to original Cramer-Shoup encryption scheme; see the full version for details). In addition, both the encryption and the decryption algorithms only require to perform operations in the first group $\mathbb{G}_1$. A detailed comparison is given in Section 7.4, where we also introduce a more efficient version of that threshold encryption scheme, for which the ciphertexts have the same size as the ciphertexts of the original Cramer-Shoup encryption scheme.

6 Pseudo-Random Projective Hash Functions and Disjunctions

In this section, we sometimes make explicit use of crs (or crs_1, or crs_2), the language parameters of the diverse vector space $\mathcal{V}$ (respectively of $\mathcal{V}_1$, and $\mathcal{V}_2$), to provide clearer definitions. We recall that we suppose there exists an algorithm $\mathsf{Setup}_{\mathsf{crs}}$ which can generate crs together with a trapdoor $\mathcal{T}_{\mathsf{crs}}$. Contrary to construction in previous sections, where $\mathcal{T}_{\mathsf{crs}} = \perp$, the security of the constructions in this section will depend on some properties of $\mathcal{T}_{\mathsf{crs}}$.

6.1 Pseudo-Randomness

Definition. An SPHF is said to be *pseudo-random*, if the hash value of a random word C in $\mathscr{L}_{\mathsf{crs}}$ looks random to an adversary only knowing the projection key hp and ignoring

```
$\mathsf{Exp}^{\texttt{ps-rnd-}b}(\mathcal{A}, \mathfrak{K})$
    $(\mathsf{crs}, \mathcal{T}_{\mathsf{crs}}) \overset{\$}{\leftarrow} \mathsf{Setup}_{\mathsf{crs}}(1^{\mathfrak{K}})$
    $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$
    $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs})$
    $C \overset{\$}{\leftarrow} \mathscr{L}_{\mathsf{crs}}$
    if $b = 0$ then
        $H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, C)$
    else $H \overset{\$}{\leftarrow} \Pi$
    return $\mathcal{A}(\mathsf{crs}, C, \mathsf{hp}, H)$
```

```
$\mathsf{Exp}^{\texttt{mixed-ps-rnd-}b}(\mathcal{A}, \mathfrak{K})$
    $(\mathsf{crs} = (\mathsf{crs}_1, \mathsf{crs}_2), (\mathcal{T}_{\mathsf{crs}_1}, \mathcal{T}_{\mathsf{crs}_2})) \overset{\$}{\leftarrow} \mathsf{Setup}_{\mathsf{crs}}(1^{\mathfrak{K}})$
    $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathsf{crs})$; $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs})$
    $C_2 \overset{\$}{\leftarrow} \mathscr{L}_{2,\mathsf{crs}_2}$
    $(C_1, \mathsf{st}) \overset{\$}{\leftarrow} \mathcal{A}(\mathsf{crs}, \mathcal{T}_{\mathsf{crs}_1}, \mathsf{hp}, C_2)$; $C \leftarrow (C_1, C_2)$
    if $b = 0$ or $C_1 \in \mathscr{L}_{1,\mathsf{crs}_1}$ then
        $H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, C)$
    else $H \overset{\$}{\leftarrow} \Pi$
    return $\mathcal{A}(\mathsf{st}, H)$
```

Fig. 4. Experiments $\mathsf{Exp}^{\texttt{ps-rnd-}b}$ and $\mathsf{Exp}^{\texttt{mixed-ps-rnd-}b}$ for pseudo-randomness and mixed pseudo-randomness

the hashing key hk and a witness for the word C. More precisely, this property is defined by the experiments $\mathsf{Exp}^{\texttt{ps-rnd-}b}$ depicted in Fig. 4. Contrary to smoothness, this property is computational. A projective hashing function which is pseudo-random is called a PrPHF. A PrPHF is not necessarily smooth.

Link with Hard Subset Membership Languages. It is easy to see that an SPHF over a hard subset membership family of languages is pseudo-random. This yields a way to create PrPHF under DDH using Example 1. However, this is inefficient since, in this case $\mathcal{X}$ has dimension 2, while we would prefer to have $\mathcal{X}$ of dimension 1. Actually, since for hard subset membership languages, $\mathscr{L}_{\mathsf{crs}} \neq \mathcal{X}$, any SPHF based on diverse vector space for these languages is such that $\mathcal{X}$ has dimension at least 2. More generally, as shown in 5.4, for a hard subset membership language based on κ-Lin, $\mathcal{X} = \mathbb{G}^{1\times(\kappa+1)}$ and $\mathscr{L}_{\mathsf{crs}}$ has dimension κ. That is why, we introduce another way to construct PrPHF, still based on diverse vector spaces, but not using hard subset membership languages.

6.2 Canonical PrPHF under κ-Lin

Let us construct a diverse vector space $(\mathcal{X}, \mathscr{L}, \mathcal{R}, \mathbb{G}, n, k, \Gamma, \theta)$ which yields a pseudo-random SPHF under κ-Lin in the cyclic group $\mathbb{G}$.

We set $\mathcal{X} = \mathscr{L}_{\mathsf{crs}} = \{\perp\}$ and $\hat{\mathcal{X}} = \hat{\mathscr{L}}_{\mathsf{crs}} = \mathbb{G}^{\kappa}$. For $\mathsf{DDH} = 1$-Lin, we get a PrPHF with $\mathcal{X}$ of dimension 1, which is the best we can do using diverse vector spaces. Even though the resulting projective hash function will be smooth, the smoothness property is completely trivial, since $\mathscr{L}_{\mathsf{crs}} \setminus \mathcal{X}$ is empty, and does not imply the pseudo-randomness property. We will therefore need to manually prove the pseudo-randomness.

The "language" is defined by $\mathsf{crs} = (\zeta_1, \ldots, \zeta_\kappa) \overset{\$}{\leftarrow} \mathbb{G}^{\kappa}$ and the PrPHF by:

$$\Gamma := \begin{pmatrix} \zeta_1 & 0 & \ldots & 0 \\ 0 & \zeta_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \zeta_\kappa \end{pmatrix} \in \mathbb{G}^{\kappa\times\kappa} \qquad \boldsymbol{\lambda} := \begin{pmatrix} \hat{\zeta}_1 \\ \hat{\zeta}_2 \\ \vdots \\ \hat{\zeta}_\kappa \end{pmatrix} \in \mathbb{Z}_p^{\kappa} \quad \theta(\perp) := \begin{pmatrix} g \\ g \\ \vdots \\ g \end{pmatrix} \in \mathbb{G}^{\kappa}$$

$$\mathsf{hk} := \boldsymbol{\alpha} \overset{\$}{\leftarrow} \mathbb{Z}_p^{1\times\kappa} \qquad \mathsf{hp} := (\gamma_1, \ldots, \gamma_\kappa)^{\intercal} = (\zeta_1^{\alpha_1}, \ldots, \zeta_\kappa^{\alpha_\kappa})^{\intercal} \in \mathbb{G}^{\kappa}$$

$$H := \prod_{i=1}^{n} g^{\alpha_i} = g^{\sum_{i=1}^{n} \alpha_i} = \prod_{i=1}^{n} \gamma_i^{\hat{\zeta}_i} =: H',$$

where $\boldsymbol{\lambda}$ is the witness for $C = \perp$, with $\zeta_i = g^{1/\hat{\zeta}_i}$. The pseudo-randomness directly comes from the hardness of κ-Lin.

6.3 Disjunction of an SPHF and a PrPHF

Let $\mathcal{V}_1 = (\mathcal{X}_1, \mathscr{L}_1, \mathcal{R}_1, \mathfrak{G}_1, n_1, k_1, \Gamma^{(1)}, \theta_1)$ and $\mathcal{V}_2 = (\mathcal{X}_2, \mathscr{L}_2, \mathcal{R}_2, \mathfrak{G}_2, n_2, k_2, \Gamma^{(2)}, \theta_2)$ be two diverse vector spaces over two multiplicatively sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. Let $\mathcal{V} = (\mathcal{X}, \mathscr{L}, \mathfrak{G}, n, k, \Gamma, \theta)$ be the vector space corresponding to the disjunction of the two previous languages. We have already seen that this vector space corresponds to a smooth projective hash function.

But, if the second language is the canonical PrPHF under κ-Lin, the smoothness brings nothing, since $\mathcal{X} = \mathscr{L}$. Therefore, we need to prove a stronger property called *mixed pseudo-randomness*.

Definition of Mixed Pseudo-Randomness. The resulting SPHF is said mixed pseudo-random, if the hash value of a word $C = (C_1, C_2)$ looks random to the adversary, when $C_1 \notin \mathscr{L}_1$ is chosen by the adversary, while C_2 is chosen at random in $\mathscr{L}_2$. More precisely, the mixed pseudo-randomness property is defined by the experiments $\mathsf{Exp}^{\texttt{mixed-ps-rnd-}b}$ depicted in Fig. 4.

Proof of Mixed Pseudo-Randomness. The proof of mixed pseudo-randomness is actually close to the one for computational soundness of trapdoor smooth projective functions in [6]. It requires that $\mathcal{T}_{\mathsf{crs}_1}$ contains enough information to be able to compute the discrete logarithm of elements of $\Gamma^{(1)}$, denoted $\mathfrak{L}(\Gamma^{(1)})$.

The proof reduces the pseudo-randomness property to the mixed pseudo-randomness property. The detailed proof is quite technical and can be found in the full version. Basically, we choose a random hashing key ε and we randomize it using a basis of the kernel of $\mathfrak{L}(\Gamma^{(1)})$ and projection keys given by the pseudo-randomness game (for some fixed word C_2, using an hybrid method). Then we show how to compute from that, a valid projection key hp for the language of the disjunction together with a hash value H of (C_1, C_2), for $C_1 \notin \mathscr{L}_1$. This value H is the correct hash value, if the hash values of C_2, given by the challenger of the hybrid pseudo-randomness game, were valid; and it is a random value, otherwise. That proves that an adversary able to break the mixed pseudo-randomness property also breaks the pseudo-randomness property.

7 One-Time Simulation-Sound NIZK from Disjunctions of an SPHF and a PrPHF

7.1 NIZK from Disjunctions of an SPHF and a PrPHF

The construction is identical to the one in Section 5.1, except that the second diverse vector space $\mathcal{V}_2$ is just supposed to be a PrPHF, and no more supposed to be related to a hard subset membership language $\mathscr{L}_2$. However, we suppose that the disjunction of $\mathcal{V}_1$

and $\mathcal{V}_2$ yields a mixed pseudo-random SPHF, which is the case if $\mathcal{T}_{\mathsf{crs}}$ contains enough information to compute the discrete logarithm of elements of $\Gamma^{(1)}$.

Completeness and zero-knowledge can be proven exactly in the same way. It remains therefore to prove the soundness property, under the mixed pseudo-randomness. The proof is very similar to the one in Section 5.1: if $\boldsymbol{\pi}$ is a proof of some word $C_1 \notin \mathscr{L}_1$, then it is possible to compute the hash value of any word (C_1, C_2) with $C_2 \in \mathscr{L}_2$ as $H' := \hat{\boldsymbol{C}}_2 \bullet \boldsymbol{\pi}$. This comes from the fact that if $C_2 \in \mathscr{L}_2$, then there exists $\boldsymbol{\lambda_2}$ such that $\hat{\boldsymbol{C}}_2 = \boldsymbol{\lambda_2} \bullet \Gamma^{(2)}$, hence:

$$H' = \boldsymbol{\lambda_2} \bullet \Gamma^{(2)} \bullet \boldsymbol{\pi} = \boldsymbol{\lambda_2} \bullet (\hat{\boldsymbol{C}}_1 \otimes \mathsf{Id}_{k_2}) \bullet \boldsymbol{\gamma^{(2)}} = (\hat{\boldsymbol{C}}_1 \otimes \boldsymbol{\lambda_2}) \bullet \boldsymbol{\gamma^{(2)}},$$

which is the hash value of (C_1, C_2) computed using $\mathsf{ProjHash}$ and witness $\boldsymbol{\lambda_2}$. But the mixed pseudo-randomness property ensures that this value looks uniformly random when C_2 is chosen randomly in $\mathscr{L}_2$. That proves the soundness property.

7.2 One-Time Simulation-Sound NIZK

Unfortunately, for the one-time simulation-sound variant, this is not as easy: the construction in Section 5.3 seems difficult (if at all possible) to prove sound. The main problem is that the security proof of mixed pseudo-randomness is not statistical, so we do not know $\mathsf{hk} = \boldsymbol{\alpha}$, but only some representation of $\boldsymbol{\alpha}$, which does not allow computing the proof $\boldsymbol{\pi'}$ of a word C_1' for a tag $\mathsf{tag}_{C_1'}$. Directly adapting the proof with a 2-smooth $\mathcal{V}_1$ would require to choose from the beginning $\boldsymbol{\pi'}$ (as is chosen hp from the beginning), but that is not possible since C_1' and tag' (the tag for C_1') are not known at the beginning of the game.

Our solution is to use the tag bit-by-bit. So we just need to guess which bit is different between tag_{C_1} and $\mathsf{tag}_{C_1'}$. This idea is inspired from [8]. Details can be found in the full version.

7.3 Concrete Instantiation and Comparison with Previous Work

If $\mathcal{V}_1$ is a diverse vector space over $\mathbb{G}_1$ (for which $\mathcal{T}_{\mathsf{crs}_1}$ gives enough information to compute the discrete logarithm of $\Gamma^{(1)}$) and $\mathcal{V}_2$ is the canonical PrPHF under DDH in Section 6.2, where $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a bilinear group where DDH is hard in $\mathbb{G}_2$, then we get an NIZK and a one-time simulation sound NIZK whose proof is composed of only $n_2 = 1$ group element in $\mathbb{G}_1$. More generally, if $\mathcal{V}_2$ is canonical PrPHF under κ-Lin, then the proof consists of only κ group elements, one less than our first construction in Section 5.4. However, this encompasses slightly fewer languages than this first construction, due to the restriction on $\mathscr{L}_1$ and $\mathcal{T}_{\mathsf{crs}_1}$. More precisely, our NIZK handles the same languages as Jutla-Roy NIZK in [20,21].

Table 1 compares NIZK for linear subspaces as Jutla and Roy call it in [20], i.e., any language over $\mathbb{G}_1$ (first group of some bilinear group) for which there exists a diverse vector space $\mathcal{V}_1$ (assuming θ is the identity function and a witness is $\boldsymbol{\lambda} \in \mathbb{Z}_p^k$). Some of the entries of this table were derived from [21] and from [26]. The DDH (in $\mathbb{G}_2$) variant requires asymmetric bilinear groups, while the κ-Lin variant for $\kappa \geq 2$ could work on symmetric bilinear groups.

Table 1. Comparison of NIZK for linear subspaces

			DDH (in $\mathbb{G}_2$)		DLin (in $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$)	
		WS	Proof $\|\pi\|$	Pairings	Proof $\|\pi\|$	Pairings
Groth-Sahai [18]			$n + 2k$	$2n(k + 2)$	$2n + 3k$	$3n(k + 3)$
Jutla-Roy [20]		✓	$n - k$	$(n - k)(k + 2)$	$2n - 2k$	$2(n - k)(k + 2)$
Libert et al. [26]					3	$2n + 4$
Libert et al. [26]	RSS				4	$2n + 6$
Libert et al. [26]	USS				20	$2n + 30$
Jutla-Roy [21]		✓	1	$n + 1$	2	$2(n + 2)$
§5.1			2	$n + 2$	3	$2n + 3$
§7.1		✓	1	$n + 1$	2	$2n + 2$
§5.3	OTSS		2	$2n + 2$	3	$4n + 3$
§7.2	OTSS	✓	1	$\nu n + 2$	2	$2\nu n + 2$

– full table with CRS sizes in the full version;
– $n = n_1$, $k = k_1$, and $\nu = 2\mathfrak{K}$; pairings: number of pairings required to verify the proof;
– sizes $|\cdot|$ are measured in term of group elements ($\mathbb{G}_1$ and $\mathbb{G}_2$, or $\mathbb{G}$ if the bilinear group is symmetric). Generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ (for DDH in $\mathbb{G}_2$) or $g \in \mathbb{G}$ (for DLin) are not counted in the CRS;
– *OTSS*: one-time simulation-soundness; *RSS*: single-theorem relative simulation-soundness [19] (weaker than OTSS); *USS*: universal simulation-soundness (stronger than OTSS);
– WS: witness-samplability in [20], generation of crs so that $\mathcal{T}_{\mathsf{crs}_1}$ enables us to compute the discrete logarithms of Γ_1. This slightly restricts the set of languages which can be handled.

First of all, as far as we know, our one-time simulation-sound NIZK is the most efficient such NIZK with a constant-size proof: the single-theorem relatively-sound construction of Libert et al. [26] is weaker than our one-time simulation-sound NIZK and requires at least one group element more in the proof, while their universal simulation-sound construction is much more inefficient. A direct application of our construction is our efficient structure-preserving threshold IND-CCA encryption scheme, under DDH.

Second, the DLin version of our NIZK in Section 5.1 is similar to the one by Libert et al. [26], but our DLin version of our NIZK in Section 7.1 is more efficient (the proof has 2 group elements instead of 3). Furthermore, the ideas of the constructions in [26] seem quite different.

Third, our NIZK in Section 7.1 is similar to the one by Jutla and Roy in [21] for DDH. However, in our opinion, our construction seems to be more modular and simpler to understand. In addition, under κ-Lin, with $\kappa \geq 2$, our construction is slightly more efficient in terms of CRS size and verification time.

7.4 Application: Threshold Cramer-Shoup-like Encryption Scheme (Variant)

In the construction of Section 5.5, we can replace the previous one-time simulation-sound NIZK by this new NIZK. This yields a threshold encryption where the ciphertext size only consists of 4 group elements as the original Cramer-Shoup encryption scheme, at the expense of having a public key size linear in the security parameter.

Our two schemes are threshold and *structure-preserving* [5]: they are "compatible" with Groth-Sahai NIZK, in the sense that we can do a Groth-Sahai NIZK to prove that we know the plaintext of a ciphertext for our encryption schemes. In addition, normal decryption does not require any pairings, which still are very costly, compared to exponentiations. A detailed comparison with existing efficient IND-CCA encryption schemes based on cyclic or bilinear groups is given in the full version. To summarize, to the best of our knowledge, our two constructions are the most efficient threshold and structure-preserving IND-CCA encryption schemes.

Acknowledgments. We would like to thank Jens Groth and the anonymous reviewers for detailed comments on a previous version of this paper. This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project, the CFM Foundation, the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet, and the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

References

1. Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D.: SPHF-friendly non-interactive commitments. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 214–234. Springer (Dec 2013)
2. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: New constructions and applications. Cryptology ePrint Archive, Report 2014/483 (2014), http://eprint.iacr.org/2014/483
3. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-based group key exchange in a constant number of rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer (Apr 2006)
4. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer (Aug 2009)
5. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer (Aug 2010)
6. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer (Aug 2013)
7. Blazy, O., Pointcheval, D., Vergnaud, D.: Round-optimal privacy-preserving protocols with smooth projective hash functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 94–111. Springer (Mar 2012)
8. Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 435–460. Springer (Aug 2013)
9. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906 (2014), http://eprint.iacr.org/2014/906
10. Coron, J.S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer (Aug 2013)

11. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer (Aug 1998)
12. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer (Apr / May 2002)
13. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer (Aug 2013)
14. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer (May 2013)
15. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 467–476. ACM Press (Jun 2013)
16. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer (May 2003), http://eprint.iacr.org/2003/032.ps.gz
17. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer (Dec 2006)
18. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer (Apr 2008)
19. Jutla, C.S., Roy, A.: Relatively-sound NIZKs and password-based key-exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 485–503. Springer (May 2012)
20. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer (Dec 2013)
21. Jutla, C.S., Roy, A.: Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 295–312. Springer (Aug 2014)
22. Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 78–95. Springer (May 2005)
23. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer (May 2001)
24. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer (Mar 2011)
25. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 289–307. Springer (Aug 2013)
26. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 514–532. Springer (May 2014)