# Verifier-Based Password-Authenticated Key Exchange: New Models and Constructions

Fabrice Benhamouda and David Pointcheval

ENS, Paris, France *

**Abstract.** While password-authenticated key exchange (or PAKE) protocols have been deeply studied, a server corruption remains the main threat, with many concrete cases nowadays. Verifier-based PAKE (or VPAKE) protocols, initially called Augmented-PAKE, have been proposed to limit the impact of any leakage. However, no satisfactory security model has ever been proposed to quantify the actual security of a protocol in the standard model. The unique model proposed so far is an ideal functionality in the universal composability (UC) framework, but is only meaningful in idealized models.
In this paper, we first enhance the Bellare-Pointcheval-Rogaway game-based model for PAKE to VPAKE protocols, and then propose the first game-based security model for both PAKE and VPAKE protocols that additionally handles related passwords. It also allows a VPAKE protocol to be secure in the standard model. We then propose several VPAKE candidates which involve smooth projective hash functions and multi-linear maps.

**Keywords.** Multi-linear maps, smooth projective hash functions, authentication, key exchange.

## 1 Introduction

Outsourced applications require strong user authentication in order to guarantee fine-grained access-control. In such situations, authenticated key exchange (AKE) protocols proved to be quite important. Since the seminal Diffie-Hellman paper [DH76] with the first key exchange protocol, though without any authentication, many variants have been proposed with various additional authentication means. The most classical one is the certificate-based authentication, that uses signatures, as in SSL and TLS. However, this relies on a public-key infrastructure to register and certify public keys. Unfortunately, such secure certification processes are not always available or reliable.

For a human being, the most convenient authentication means is definitely a simple password which can be easily memorized, and which does not require anything else to be checked. This scenario has first been proposed by Bellovin and Merritt [BM92] in 1992, with the famous Encrypted Key Exchange (EKE) protocol. However, as it has been recently illustrated (50 million LivingSocial passwords stolen in April 2012, more than 6 million LinkedIn passwords stolen in June 2012, or data, including passwords, for millions of Adobe customers stolen earlier this October), servers can be hacked and the authentication means stolen. When the passwords are stored in clear on the server, this can be dramatic, especially since users often use related passwords (if not the same) for many providers. To overcome this issue, Bellovin and Merritt also proposed the Augmented EKE protocol [BM93], where the server just stores a means, called a *verifier*, to verify that the client used the correct password, but not the password itself. Verifiers are usually hash values of passwords with a salt. This limits the impact of leakage of information on the server side, since,while it does not prevent *off-line dictionary attacks*, it forces the adversary to spend a lot of time to learn many passwords. This gives enough time for letting the users renew their passwords.

Of course, security is always limited when authentication relies on such a low-entropy secret, as a password: an attacker can make as many attempts as he wants with various passwords until he finds the correct one. This is the so-called *on-line dictionary attack*, which cannot be avoided. Usual passwords have a quite small entropy and thus the number of trials before finding the correct password might not be so high. But the impact can be limited by refusing connection attempts after a few failures or by increasing the waiting time before a new attempt. As a consequence, with appropriate restrictions, on-line dictionary attacks can be mitigated. Therefore, we are left with the

* CNRS – UMR 8548 and INRIA – EPI Cascade

problem of designing password-authenticated key exchange (PAKE) protocols guaranteeing that on-line dictionary attacks are the best attacks an adversary can mount. This means that an active attack should not help him to eliminate more than one password from the list of potential passwords, and a passive attack should not leak any information at all. When the server just stores verifiers for the passwords, such protocols are called verifier-based password-authenticated key exchange (VPAKE) protocols.

### 1.1 Related Work

PAKE protocols have been introduced by Bellovin and Merritt [BM92]. They allow two players to agree on a common session key while using a simple password (a low-entropy secret) as authentication means. Bellare, Pointcheval and Rogaway [BPR00] proposed the first game-based security model (BPR), and several security results have been proven: EKE in the ideal-cipher model [BCP03], and then in the random-oracle model [BCP04, AP05]. Katz, Ostrovsky and Yung [KOY01] designed the first efficient scheme (KOY), provably secure in the standard model, later generalized by the Gennaro and Lindell (GL) framework [GL03].

The main drawback of the BPR security model is the strong assumption on the password distribution: passwords are assumed to be independently and uniformly distributed, whereas human beings use biased distributions, and definitely related passwords. Another major issue is the find-then-guess game instead of a real-or-random game. While they are polynomially equivalent [BDJR97] using an hybrid technique, this polynomial loss can make a huge difference in the password-based setting where the success probability of an adversary is not negligible but linear in $2^{-n}$, with $n$ the bit-length of the passwords. Hence, we use the stronger security model with the real-or-random flavor [AFP05] with a tight bound on the success probability of the adversary.

In 2005, Canetti *et al.* [CHK+05] managed to remove the assumption on the distribution on the passwords with an ideal functionality in the UC framework [Can01]. They extended the GL framework, making use of smooth projective hash functions [CS02], to construct efficient concrete schemes. More constructions have thereafter been proposed, with adaptive security and/or in only one round [ACP09, KV11, BBC+13b, ABB+13].

Regarding VPAKE, while many protocols have been designed and informally analyzed, no game-based security model has been proposed so far, but just quite recently an ideal functionality in the UC framework [GMR06] with a generic conversion. To model an off-line dictionnary attack, an ideal query (the `OfflineTestPwd` query) is made available to the adversary. This ideal query enables to count the number of passwords tested off-line, as the `TestPwd` ideal query enables to count the number of passwords tested via an on-line dictionnary attack (in UC-PAKE and UC-VPAKE). Unfortunately, because of the `OfflineTestPwd` query, such a functionality can only be realized in an idealized model, such as the random oracle, ideal cipher or generic model. Actually, the generic construction proposed by the authors is in the random-oracle model.

### 1.2 Contributions

In this paper, we deal with the two above problems: related passwords for PAKE and VPAKE, and possible instantiations of VPAKE protocols in the standard model. More precisely, after a simple enhancement of the BPR security model to VPAKE, we present a new game-based security model for PAKE and VPAKE that handles related passwords, with the real-or-random flavor. We could show that most of the well-known previous PAKE protocols are actually secure in this stronger security model. The main advantage of this security model for VPAKE is that it does not require the proofs to use an idealized model. We actually provide an illustration with a VPAKE secure in the standard model. While it requires a non-standard assumption, we provide evidences of its validity in the generic model.

To this aim, we define the notion of *password hashing* to derive the verifier from the password and a salt, with two constructions. One of them involves multi-linear maps in a similar way as in [BR13], but with a new proof technique in the generic model because of the tight reduction required for proving the complexity of an off-line dictionary attack.

This immediately leads to VPAKE protocols secure in our model. These constructions make use of smooth projective hash functions (SPHFs), as in [BBC+13a, BBC+13b]. Some of the SPHFs used are based on multi-linear maps or are for systems of quadratic pairing equations over ciphertexts, as detailed in [BBC+13c]. This is the first concrete use of these more advanced SPHFs for a concrete and somehow efficient application.

# 2 Preliminaries

## 2.1 Notations

In this paper, the qualities of an adversary $\mathcal{A}$ against a security notion $\mathtt{sec}$ is measured by his success or his advantage in certain experiments $\mathsf{Exp}_{\mathsf{sec}}$ or games $\mathbf{G}_{\mathsf{sec}}$, denoted by $\mathsf{Succ}_{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}_{\mathsf{sec}}(\mathcal{A}, \mathfrak{K}) = 2 \cdot \mathsf{Succ}_{\mathsf{sec}}(\mathcal{A}, \mathfrak{K}) - 1$ respectively, where $\mathfrak{K}$ is the security parameter. We denote by $\mathrm{negl}(\mathfrak{K})$ a quantity which is $O(1/\mathfrak{K}^k)$ for any $k \geq 1$ and we denote by $\xleftarrow{\$}$ the outcome of a probabilistic algorithm or the sampling from a uniform distribution. Finally, $\{0,1\}^n$ is the set of bit-strings of length $n$, or $n$-bit-strings. If $x \in \{0,1\}^n$ is such a bit-string, $x[i]$ denotes the $i$-th bit of $x$.

## 2.2 Multi-Linear Maps and Graded Rings

In the core of this paper, we use an idealized version of asymmetric multi-linear maps with the notations introduced in [BBC+13c]. Very roughly, an $n$-graded ring is just the extension of an asymmetric bilinear map $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ from 2 groups ($\mathbb{G}_1$ and $\mathbb{G}_2$), to $n$ groups, denoted by $\mathfrak{G}_{\boldsymbol{e}_1}, \ldots, \mathfrak{G}_{\boldsymbol{e}_n}$. Contrary to asymmetric bilinear groups, the group law is denoted by $\oplus$ instead of $\cdot$, and the pairing operation is denoted by $\cdot \odot \cdot$ instead of $e(\cdot, \cdot)$. As for asymmetric bilinear groups, there are some restrictions on the operands for $\odot$. For the sake of simplicity, graded rings are supposed to be of prime order $p$, even if, in reality, it is not exactly the case.

Formal definitions of graded rings can be found in Appendix A.1, together with a discussion on how to implement our schemes using the multi-linear map construction of Coron, Lepoint and Tibouchi [CLT13][1]. It is highly recommended to read this appendix before reading the other appendices dealing with graded rings.

# 3 Password Hashing

## 3.1 Definitions

A password hashing scheme formalizes the way the salt and the hash value are generated in order to allow password verification on a server, so that the values stored on the server-side leak as little information as possible on the password. Basically, the hash value (or verifier) results from a one-way process on a salt and the password.

[1] The first construction of multi-linear maps of Garg, Gentry and Halevi [GGH13] does not suit our purpose because the GXDH assumption does not hold. And the trick of Brakerski and Rothblum [BR13] consisting in not giving all parameters to the adversary is not possible in our case, since the adversary has to be able to sample level-0 values and to encrypt and randomize.

**Password Hashing Scheme.** It is defined by four algorithms:
- $\mathsf{Setup}(1^{\mathfrak{K}}, 1^n)$ is a probabilistic algorithm which takes as input the security parameter $\mathfrak{K}$ and the bit-length $n \leq \mathfrak{K}$ of the password, and outputs parameters $\mathsf{param}$;
- $\mathsf{PSalt}(\mathsf{param})$ is a probabilistic algorithm which takes as input the parameters $\mathsf{param}$ and outputs a salt $s$ in some set $\mathbb{S}$ (depending on $\mathsf{param}$); we just require that checking that some bit-string is in $\mathbb{S}$ or not can be done in polynomial time (in $\mathfrak{K}$);
- $\mathsf{PPreHash}(\mathsf{param}, \pi)$ is a deterministic algorithm which takes as input the parameters $\mathsf{param}$ and a password $\pi \in \{0,1\}^n$, and outputs a pre-hash value $P$;
- $\mathsf{PHash}(\mathsf{param}, s, P)$ is a deterministic algorithm which takes as input the parameters $\mathsf{param}$, a salt $s$ and a pre-hash value $P$, and outputs a hash value $H$.

We also define $\mathsf{PFullHash}(\mathsf{param}, s, \pi) = \mathsf{PHash}(\mathsf{param}, s, \mathsf{PPreHash}(\mathsf{param}, \pi))$. For convenience, we will sometimes omit $\mathsf{param}$ as argument.

The $\mathsf{PPreHash}$ algorithm might look artificial, but it helps to move the algebraic part of the hashing, compatible with $\mathsf{SPHF}$s and the GL framework, in the $\mathsf{PHash}$ algorithm. In this paper, the $\mathsf{PPreHash}$ algorithm either corresponds to the identity function ($P = \pi$), or is used to make the password random-looking by applying a random oracle.

We remark that our definition supposes passwords are $n$-bit-strings. This is often not the case in practice, if passwords are ASCII strings containing only alphanumeric caracters. But it is possible to encode such ASCII strings into $n$-bit-strings, with $n$ close to the logarithm of the number of such strings.

**Security Properties.** In order to be applied in $\mathsf{VPAKE}$, a password hashing scheme has to satisfy four properties:
- *Second pre-image resistance.* For any password $\pi \in \{0,1\}^n$, and any adversary $\mathcal{A}$ running in polynomial time in $\mathfrak{K}$:

$$\mathsf{Adv}_{\mathtt{snd}}(\mathcal{A}, \mathfrak{K}) \coloneqq \Pr\Big[\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}, 1^n)\ ;\ s \xleftarrow{\$} \mathsf{PSalt}(\mathsf{param}, \pi)\ ;\ P \leftarrow \mathsf{PPreHash}(\mathsf{param}, \pi)\ ;$$
$$P' \xleftarrow{\$} \mathcal{A}(\mathsf{param}, P, s)\ ;\ P' \neq P \text{ and } \mathsf{PHash}(\mathsf{param}, s, P) = \mathsf{PHash}(\mathsf{param}, s, P')\Big] = \mathrm{negl}(\mathfrak{K});$$

- *Entropy preservation.* For any password distribution[2] $\mathcal{D}$ of min-entropy $\beta$, and any adversary $\mathcal{A}$ running in polynomial time in $\mathfrak{K}$:

$$\mathsf{Adv}_{\mathtt{entropy}}(\mathcal{A}, \mathfrak{K}) \coloneqq \Pr\Big[\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}, 1^n)\ ;\ (s, H) \xleftarrow{\$} \mathcal{A}(\mathsf{param})\ ;\ \pi \xleftarrow{\$} \mathcal{D}\ ;$$
$$s \in \mathbb{S} \text{ and } H = \mathsf{PFullHash}(\mathsf{param}, s, \pi)\Big] \leq 2^{-\beta} + \mathrm{negl}(\mathfrak{K})$$

- *Pre-hash entropy preservation.* For any password distribution $\mathcal{D}$ of min-entropy $\beta$, and any adversary $\mathcal{A}$ running in polynomial time in $\mathfrak{K}$:

$$\mathsf{Adv}_{\mathtt{pre\text{-}entropy}}(\mathcal{A}, \mathfrak{K}) \coloneqq \Pr\Big[\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}, 1^n)\ ;\ P \xleftarrow{\$} \mathcal{A}(\mathsf{param})\ ;\ \pi \xleftarrow{\$} \mathcal{D}\ ;$$
$$P = \mathsf{PPreHash}(\mathsf{param}, \pi)\Big] \leq 2^{-\beta} + \mathrm{negl}(\mathfrak{K})$$

- *Tight one-wayness.* Inverting $\mathsf{PHash}$ takes approximately as long as evaluating $2^\beta$ times the function $\mathsf{PHash}$, on average (i.e., brute-forcing the password), up to some small multiplicative constant, where $\beta$ denotes the min-entropy of the password distribution $\mathcal{D}$. More precisely, let us consider the game $\mathbf{G}_{\mathtt{one\text{-}way}}$ of Figure 1. The oracle **Initialize** first generates parameters and sends them back to the adversary $\mathcal{A}$. Then the adversary $\mathcal{A}$ asks queries to the oracle **Hash** to get the hash values of some random passwords (and with random salts), as many times as it wants. At the end of the game, the adversary $\mathcal{A}$ has to call the oracle **Finalize** with a pre-hash guess for one of the

[2] In this paper, all distributions are supposed to be polynomial-time samplable.

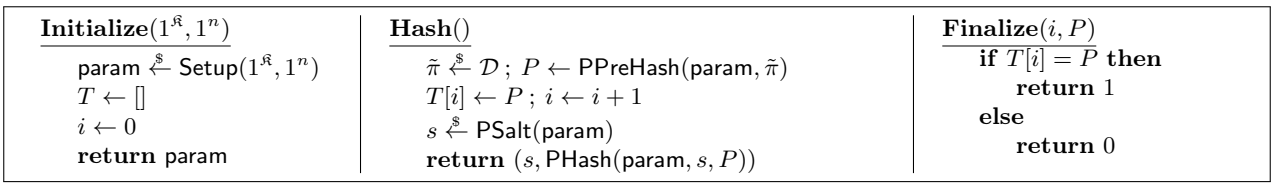


Fig. 1: Game $\mathbf{G}_{\mathsf{one\text{-}way}}$ for the one-wayness of password hashing schemes

hash values (and the corresponding salt) output by **Hash**. If the guess is correct, he wins. The password hashing scheme is tightly one-way, if for any adversary $\mathcal{A}$ running in time at most $t$, for any password distribution $\mathcal{D}$ of min-entropy $\beta$:

$$\mathsf{Succ}_{\mathsf{one\text{-}way}}(\mathcal{A}, q, \mathfrak{K}) \lesssim \frac{t}{2^{\beta} \cdot t_{\mathsf{PHash}}} + \mathrm{negl}(\mathfrak{K}),$$

where $t_{\mathsf{PHash}}$ is the time of the algorithm PHash and "$\lesssim$" means less than up to some small multiplicative constant.

We insist on the fact that the adversary can query the **Hash** oracle as many times as he wants and that it should not help him to go faster than brute-force. This would obviously not be possible without salt since a PHash evaluation would then target all the hash values.

The first three properties are actually (at least implicitly) expected for all hash functions, and are always straigthforward in our case. The technical part is to design a scheme which verifies the tight one-wayness and which is additionally compatible with the GL framework (or more precisely with an SPHF). Due to this second requirement, PHash needs some algebraic properties.

**Idealized Models.** In practice however, it seems quite difficult to achieve the tight one-wayness. One approach makes use of idealized models. For example, in the random oracle model, instead of considering the running time $t$ of the adversary and the time $t_{\mathsf{PHash}}$ of a call to PHash, we consider $q_{\mathcal{H}}$ the number of adversarial queries to the random oracle and $q_{\mathsf{PHash}}$ the number of queries to the random oracle performed by PHash itself. It seems fair to consider that if we instantiate the random oracle by a hash function such as SHA-1, in a tightly one-way hashing password scheme for the number of queries to the random oracle, we would get a tightly one-way hashing password scheme (for the original definition).

This definition in the random oracle model can be extended to other idealized models such as the generic group model, in which case, we use the number of queries to the group law instead of the time.

### 3.2 Construction Based on Random Oracles

**A Naive Password Hashing Scheme.** Let us first exhibit a trivial construction using a random oracle $\mathcal{H}$ with values in $\{0,1\}^{2\mathfrak{K}}$. For this construction, $\mathsf{param} = 1^{\mathfrak{K}}$, $\mathsf{PSalt}(\mathsf{param})$ outputs a random salt $s \in \mathbb{S} = \{0,1\}^{2\mathfrak{K}}$, $P = \mathsf{PPreHash}(\pi) = \pi$ and $H = \mathsf{PHash}(s, \pi) = \mathcal{H}(s, \pi)$. Let us show that this scheme is a password hashing scheme for any $n \leq \mathfrak{K}$:

- *Second Pre-Image Resistance.* For any password $\pi \in \{0,1\}^n$ and any salt $s$, the probability there exists another password $\pi' \in \{0,1\}^n$ such that $\mathcal{H}(s, \pi) = \mathcal{H}(s, \pi')$ is less than $2^n/2^{2\mathfrak{K}} \leq 1/2^{\mathfrak{K}}$, since $\mathcal{H}$ has values in $\{0,1\}^{2\mathfrak{K}}$, so the second pre-image resistance statistically holds.
- *Entropy Preservation.* We remark that in the entropy preservation experiment, if the adversary makes $q_{\mathcal{H}}$ queries to the random oracle, the probability of a collision is less than $q_{\mathcal{H}}^2/2^{2\mathfrak{K}}$. Let us now suppose that there are no collision. We denote by $s$ the salt and by $H$ the hash value returned by the adversary. Let $\pi \xleftarrow{\$} \mathcal{D}$. Two situations appear:
  - either $H$ is not an answer to any pair $(s, x)$ asked by the adversary to the random oracle. Then, either $(s, \pi)$ has been asked by the adversary to the random oracle, in which case $\mathcal{H}(s, \pi) \neq H$, or $(s, \pi)$ has never been asked to the random oracle, and so $\mathcal{H}(s, \pi)$ is a random string in $\{0,1\}^{2\mathfrak{K}}$ and is equal to $H$ with probability $1/2^{2\mathfrak{K}}$;

- or $H$ is the answer to $\mathcal{H}(s,x)$. Then, the probability that $\pi = x$, is at most $2^{-\beta}$, and if $\pi \neq x$, the probability $\mathcal{H}(s,\pi) = H$ is at most $1/2^{2\mathfrak{K}}$, as in the second case above.

Therefore, we get $\mathsf{Adv}_{\mathtt{entropy}}(\mathcal{A},\mathfrak{K}) \leq 2^{-\beta} + (q_{\mathcal{H}}^2+1) \times 2^{-2\mathfrak{K}}$.

– *Pre-Hash Entropy Preservation.* It is trivial since $\mathsf{PPreHash}$ is the identity function.

– *Tight One-Wayness.* We remark that the probability that two salts created by **Hash** are equal is less than $q_{\mathbf{Hash}}^2/2^{2\mathfrak{K}}$ (with $q_{\mathbf{Hash}}$ the numbers of queries to **Hash**). When there is no collision, each random oracle query enables to check at most one password for only one salt returned by **Hash** (or in other words for one index $i$ of the array $T$ in the game $\mathbf{G}_{\mathtt{one\text{-}way}}$). So we have $\mathsf{Succ}_{\mathtt{one\text{-}way}}(\mathcal{A},\mathfrak{K}) \leq q_{\mathcal{H}} \times 2^{-\beta} + q_{\mathbf{Hash}}^2 \times 2^{-2\mathfrak{K}}$, because each password appears with probability at most $2^{-\beta}$.

**An Algebraic Password Hashing Scheme.** Unfortunately, the previous construction cannot be easily used in our $\mathsf{VPAKE}$ scheme, because $\mathsf{PHash}$ has no algebraic property and so seems not compatible with an $\mathsf{SPHF}$. Let us now introduce another construction which is used later in one of our $\mathsf{VPAKE}$ schemes. Let $\mathsf{param} = \mathbb{G}$ be a (multiplicative) cyclic group of order $p$ (a prime number with more than $2\mathfrak{K}$ bits) and $\mathcal{H}$ be a random oracle with values in $\mathbb{Z}_p$. Then $\mathsf{PSalt}(\mathsf{param})$ just picks a random $a \in \mathbb{S} = \mathbb{G} \setminus \{1\}$ and outputs $s = a$, while $P = \mathsf{PPreHash}(\pi) = \mathcal{H}(\pi)$ and $H = \mathsf{PHash}(a,P) = a^P$. Let us now prove the security of the construction:

– *Second Pre-Image Resistance.* For any pre-hash values $P, P' \in \mathbb{Z}_p$ and any salt $s = a \in \mathbb{S}$, if $\mathsf{PHash}(a,P) = \mathsf{PHash}(a,P')$, $a^P = a^{P'}$ and so $P = P'$, since $a$ a generator of order $p$. So the second pre-image resistance holds perfectly.

– *Entropy Preservation and Pre-Hash Entropy Preservation.* The entropy preservation and the pre-hash entropy preservation can be proven exactly as the entropy preservation for the previous scheme, except $\mathcal{H}(s,x)$ and $\mathcal{H}(s,\pi)$ are replaced by $s^{\mathcal{H}(x)}$ and $s^{\mathcal{H}(\pi)}$ for entropy preservation, and by $\mathcal{H}(x)$ and $\mathcal{H}(\pi)$ for pre-hash entropy preservation.

– *Tight One-Wayness.* To prove this property, counting the queries to the random oracle is not sufficient: to illustrate the difficulty of the proof, let us suppose $\mathcal{D}$ is the uniform distribution over $\{0,1\}^n$. Then, for $\mu = \lceil 2^{n/2} \rceil$, the adversary can query $\mathcal{H}$ on $\mu$ arbitrary passwords $\pi'_1, \ldots, \pi'_\mu \in \{0,1\}^n$ to get $\mu$ pre-hash values $P'_1, \ldots, P'_\mu \in \mathbb{Z}_p$ and do $\mu$ requests to **Hash** and gets $\mu$ salts $s_1, \ldots, s_\mu$ and $\mu$ hash values $H_1 = \mathsf{PFullHash}(s_1,\pi_1), \ldots, H_\mu = \mathsf{PFullHash}(s_\mu, pw_\mu)$. Then with constant probability, there is a collision between the set $\{\pi_1, \ldots, \pi_\mu\}$ and the set $\{\pi'_1, \ldots, \pi'_\mu\}$, and so there exists $i,j$ such that, $\mathsf{PHash}(s_i, P'_j) = H_i$. This attack breaks the one-wayness with only about $2^{n/2}$ queries to the random oracle. But one can remark that such an adversary makes about $\mu^2 \approx 2^n$ evaluations of $\mathsf{PHash}(s_i, P_j)$, and thus $2^n$ exponentiations.
  In Appendix C.1, we prove that any adversary has to do at least about $2^\beta$ operations in the generic group model, for any $n$ such that $2^{4n+1} < p$ (i.e., $n \lesssim \mathfrak{K}/2$). This proves the tight one-wayness since the cost of evaluating $\mathsf{PFullHash}$ is one exponentiation and one evaluation of the random oracle (which is anyway faster). This result can be seen as an extension of Theorem 2 from [Sch01]. However, we give the full proof in the appendix because the original proof was not rigorous enough for our purpose.

*Remark on the Use of the Generic Model.* One way to avoid using the generic group model would be to use a salt $s' \in \{0,1\}^{2\mathfrak{K}}$ in $\mathsf{PPreHash}$: $P = \mathcal{H}(s',\pi)$. But this is not supported by our password hashing schemes, and would require the server to send the salt to the client at the beginning of our generic $\mathsf{VPAKE}$ protocol in Section 5.3, in an additional flow. In practice, since a $\mathsf{VPAKE}$ protocol is usually initiated by a client, this would add another pre-flow from the client to the server, which would make a four-round protocol. Therefore we prefer to forbid the use of a salt in $\mathsf{PPreHash}$ at the cost of relying on the generic model for this password hashing scheme.

### 3.3 Construction Based on Multi-Linear Maps

Let us now introduce a completely algebraic construction based on multi-linear maps. Before showing our actual construction, let us first give some failed attempts at constructing a password hashing scheme in a cyclic group, to show that constructing such a fully algebraic password hashing scheme is not as straightforward as it may seem.

**Failed Attempts.** Let $\mathbb{G}$ be a cyclic group. A first straightforward (incorrect) construction would be, for $\mathsf{PSalt}(\mathsf{param})$ to output a random salt $s = a \in \mathbb{S} = \mathbb{G} \setminus \{1\}$ and for $\mathsf{PHash}(a, \pi)$ to output $H = a^\pi$. But, generic algorithms such as baby-step giant-step [Sha71] or Pollard's lambda algorithm [Pol00] enable to invert this function with about $2^{n/2}$ group operations which is far lower than $2^n$.

A second idea would be to use $n$ group elements $(a_1, \ldots, a_n) \in \mathbb{G}^n$ as salt $s$ and to set $H = a_1^{\pi[1]} \cdots a_n^{\pi[n]}$. But the baby-step giant-step algorithm still works here.

**Construction Based on Multi-Linear Maps.** Let us now consider an $(n+1)$-graded ring $\mathfrak{G}$, and let $(a_{i,b})_{i,b}$ (with $1 \le i \le n$ and $b \in \{0,1\}$) be random non-nul group elements with $a_{i,b} \in \mathfrak{G}_{\boldsymbol{e}_i} \setminus \{0\}$. Parameters are $\mathsf{param} = (\mathfrak{G}, (a_{i,b})_{i,b})$. A salt is a random non-nul group element $s \in \mathbb{S} = \mathfrak{G}_{\boldsymbol{e}_{n+1}} \setminus \{0\}$, while $P = \mathsf{PPreHash}(\pi) = \pi$ and $H = \mathsf{PHash}(s, \pi) = a_{1,\pi[1]} \odot \cdots \odot a_{n,\pi[n]} \odot s$. This construction can actually be seen as a particular case of the construction of Brakerski and Rothblum in [BR13] without wildcards, and with an additive salt $s$. Let us now prove the security of this construction:

- *Second Pre-Image Resistance and Entropy Preservation.* For any pair of distinct passwords (or pre-hash values) $P = \pi$ and $P' = \pi'$, the probability (over the choice of $(a_{i,b})$) that $a_{1,\pi[1]} \odot \cdots \odot a_{n,\pi[n]} = a_{1,\pi'[1]} \odot \cdots \odot a_{n,\pi'[n]}$ is $1/(p-1)$, because there exists $i$ such that $\pi[i] \neq \pi'[i]$ and $a_{i,\pi[1]}$ and $a_{i,\pi'[i]}$ are independent and uniformly random in $\mathfrak{G}_{\boldsymbol{e}_i} \setminus \{0\}$. Therefore, with probability at least $\frac{n(n+1)}{2(p-1)}$, the function $\pi \in \{0,1\}^n \mapsto a_{1,\pi[1]} \odot \cdots \odot a_{n,\pi[n]}$ is injective, and so is $\pi \mapsto \mathsf{PHash}(s, \pi)$ (for $s \in \mathbb{S}$). Hence, the second pre-image resistance and the entropy preservation are verified statistically.
- *Pre-Hash Entropy Preservation.* It is trivial since $\mathsf{PPreHash}$ is the identity function.
- *Tight One-Wayness.* First, we remark that this property cannot be proven as in [BR13], because their proof is not tight as we understand it, since the security model is quite different. We therefore need to do a completely new proof from scratch, which is also of independent interest.
  In the graded ring generic model, each element is represented by a random string, and operations $\oplus$ and $\odot$ are performed by two oracles. Our goal in the proof is to be able to show that we can simulate these oracles without knowing the real passwords, and that the adversary's advantage at distinguishing the real oracles and the simulated ones is at most $t/2^n$, with $t$ the number of queries to the oracles (or multi-linear group operations).
  Let us assume that $(a_{i,b})_{i,b}$ are random group elements such that $a_{i,b} \in \mathfrak{G}_{\boldsymbol{e}_i}$, without the restriction $a_{i,b} \neq 0$. This new distribution of $(a_{i,b})_{i,b}$ is statistically indistinguishable from the original one. Let $\tilde{\pi}_1, \ldots, \tilde{\pi}_q$ be the $q = q_{\mathbf{Hash}}$ passwords used in the $q$ queries to **Hash**, and let $s_1, \ldots, s_q$ and $H_1, \ldots, H_q$ be the associated salts and hash values respectively. The passwords $\tilde{\pi}_1, \ldots, \tilde{\pi}_q$ are independently drawn from a distribution $\mathcal{D}$ of min-entropy $\beta$. The basic idea is to remark that the adversary queries to the oracles can be seen as polynomials:

$$Q_j \coloneqq \bigoplus_{\substack{k=1,\ldots,q \\ I \subset \{1,\ldots,n\},\, \pi \in \{0,1\}^n}} \left( \gamma_{k,j,I,\pi} \odot \bigodot_{i \in I} a_{i,\pi[i]} \odot s_k \right) \oplus \bigoplus_{k=1,\ldots,q} \left( \gamma_{j,k,H} \odot H_k \right),$$

  with unknowns $(a_{i,b})_{i,b}$, $(s_k)_k$ and $(H_k)_k$, and with coefficients $(\gamma_{j,k,I,\pi})_{j,k,I,\pi}$ and $(\gamma_{j,k,H})_{j,k}$. To simulate the oracles, we just return an independent random string for each of these polynomials, as if all the $H_k$'s, the $s_k$'s and the $a_{i,b}$'s were independent. We just need to return the same random string for two equal polynomials, which can be done as in [BR13], by testing whether the difference between two polynomials is 0 or not, by evaluating it at some random point.

From now on, we can assume that all the previous polynomials are formally distinct. To prove that our simulation is indistinguishable from the real oracles, it remains to show that, with probability at most $t \times 2^{-\beta}$, all these polynomials are still distinct, when $H_k$ is replaced by $a_{1,\tilde{\pi}_k[1]} \odot \cdots \odot a_{n,\tilde{\pi}_k[n]} \odot s_k$. Let us first focus on the case $q = 1$ and consider the graph whose nodes are the polynomials $Q_j$. There is an edge labeled $\pi$ between $Q_j$ and $Q_{j'}$ with $j \neq j'$ if and only if $Q_j \ominus Q_{j'} = c_{j,j'} \odot (a_{1,\pi[1]} \odot \cdots \odot a_{n,\pi[n]} \odot s_1 \ominus H_1)$ with $c_{j,j'}$ a non-zero constant, i.e., when $Q_j$ and $Q_{j'}$ would be equal if $H_1$ were the hash value of $\pi$. We remark that, with high probability, the adversary can distinguish the simulated oracles from the real oracles if and only if the real password $\tilde{\pi}_1$ labels an edge of this graph. Since $\tilde{\pi}_1$ is never used, the adversary's advantage is approximatively at most $m/2^{\beta}$, with $m$ the number of distinct labels in the edges of the graph.
To bound this number of distinct labels, we first arbitrarily remove edges in such a way that there remains only one edge per distinct label in the initial graph (and thus all edges have distinct labels). Let us now proceed by contradiction: we assume there is a cycle $Q_{j_1}, \ldots, Q_{j_\ell}, Q_{j_{\ell+1}} = Q_{j_1}$ with edges with distinct labels $\pi_1, \ldots, \pi_\ell$ in this new graph. We get:

$$0 = \bigoplus_{u=1}^{\ell} Q_{j_u} \ominus Q_{j_{u+1}} = \bigoplus_{u=1}^{\ell} c_{j_u,j_{u+1}} \odot (a_{1,\pi_u[1]} \odot \cdots \odot a_{n,\pi_u[n]} \odot s_1 \ominus H_1),$$

which is obviously impossible since all the labels $\pi_1, \ldots, \pi_\ell$ are distinct and so all the monomials $a_{1,\pi_k[1]} \odot \cdots \odot a_{n,\pi_k[n]} \odot s_1$ (for $k = 1, \ldots, \ell$) are distinct. Therefore, the new graph has no cycle and so its number of edges (and so its number $m$ of labels) is at most its number of nodes, which is at most the number of polynomials, and so at most the number $t$ of oracle queries or group operations. Thus, the adversary's advantage is approximatively at most $t/2^{\beta}$.
In Appendix C.2, we analyse the case $q \geq 2$ and we show that the adversary's advantage is approximatively at most $2t/2^{\beta}$, in all cases. This concludes the proof.

# 4 Security Models

In this section, we introduce our new game-based security model for VPAKE protocols, associated with a password hashing scheme. We first present an extension of the BPR PAKE security model [BPR00], but with the real-or-random flavor [AFP05]. We then improve it to handle related passwords.

## 4.1 A BPR-like Security Model

Let us consider a password hashing scheme $\mathsf{PH} = (\mathsf{Setup}, \mathsf{PSalt}, \mathsf{PPreHash}, \mathsf{PHash})$.

**Users and Passwords.** In this model, we consider a list of $Q$ pairs of matching client-server. Clients are denoted by $C \in \mathcal{C}$ and servers are denoted by $S \in \mathcal{S}$. Each client $C \in \mathcal{C}$ holds a password $\pi_C$, while each server $S \in \mathcal{S}$ holds a random salt $s_S \xleftarrow{\$} \mathsf{PSalt}(\mathsf{param})$ and a hash value $H_S = \mathsf{PFullHash}(\mathsf{param}, s_S, \pi_S)$ of some password $\pi_S$. We restrict ourselves to the case where for each pair $(C, S) \in Q$, $\pi_C = \pi_S$ and this common password is drawn from a distribution $\mathcal{D}$ (independently from the passwords of the other pairs). Each client or server appears exactly in one pair.

**Protocol Execution.** The adversary $\mathcal{A}$ can create several concurrent instances $U^i$ of each user $U \in \mathcal{C} \cup \mathcal{S}$, and can interact with them via the following oracle queries:
- $\mathtt{Execute}(C^i, S^j)$: this query models a passive attack in which the adversary eavesdrops on honest executions between a client instance $C^i$ and a server instance $S^j$. The output of this query consists of the messages that are exchanged during an honest execution of the protocol between $C^i$ and $S^j$ (i.e., the transcript of the protocol);

– $\texttt{Send}(U^i, U'^j, m)$: this query models an active attack, in which the adversary may intercept a message and modify it, create a new message, or simply replay or forward an existing message, to the user instance $U'^j$ in the name of the user instance $U^i$. The output of this query is the message that $U'^j$ would generate after receiving $m$. A specific message `Start` can be sent to a client, in the name of a server, to initiate a session between this client and this server.
– $\texttt{Corrupt}(S)$: this query models the server corruption. The output of this query is the salt $s_S$ and the hash value $H_S$. Any client $C_i$ with password $\pi_S$ is said to have his password hash corrupted.

For this model, as in all BPR-like models, `Execute` queries and `Send` queries can only be performed between a client and a server of the same pair, and so with the same password.

**Partnering.** Before actually defining the secrecy of the session key, and thus implicit authentication, we need to introduce the notion of partnering: Two instances are partnered if they have matching transcripts, which means that for one user its view is a part of the view of the other user. One should note that the last flow can be dropped by the adversary, without letting the sender know. The sender of this last flow thus thinks that the receiver got the message and still computes the session key.

**Security.** To actually define the semantic security of a VPAKE scheme, the adversary $\mathcal{A}$ has access to a challenge oracle $\texttt{Test}(U^i)$, available many times, in the real-or-random flavor. It slightly differs from original definitions [BPR00, AFP05], since clients and servers are paired in our model, whereas in previous models there was a unique global server with specific common passwords with every client. A random bit $b$ is chosen at the beginning of the game $\mathbf{G}$, and `Test` queries for some user instance $U^i$ are then answered as follows:
– if no session key has been computed by $U^i$, or if $U^i$ is a partnered client instance whose password hash has been corrupted, then return $\perp$;
– otherwise, return the session key of $U^i$ if $b = 1$, and a random session key if $b = 0$.
At the end of the game, the adversary $\mathcal{A}$ has to output a bit $b'$. The success probability $\mathsf{Succ}$ of $\mathcal{A}$ is the probability that $b' = b$, while its advantage is defined by $\mathsf{Adv} = 2 \cdot \mathsf{Succ} - 1$. This characterizes his ability to distinguish random keys from real keys in all *non-trivial* cases. This is the reason why all the trivial cases (for which the adversary may already know the answer: no key, or corrupted password), the `Test` query is answered by $\perp$.

A VPAKE could be considered secure if the advantage of any adversary $\mathcal{A}$, running in time $t$, in the previous experiment is bounded by:
– $q_s \times 2^{-m} + \mathrm{negl}(\mathfrak{K})$, if the adversary did not ask a `Test` query for a client with a corrupted password hash;
– $q_s \times 2^{-m} + \mathsf{Adv}_{\texttt{one-way}}(\mathcal{B}, \mathfrak{K}) + \mathrm{negl}(\mathfrak{K})$, otherwise, for some adversary $\mathcal{B}$ running in time about at most $t$,
where $q_s$ is the number of active sessions (handled with `Send` queries). Intuitively this means that to win, the adversary either has to corrupt a server and do a brute-force attack to find the pre-hash value or the password from the hash value stored in the server (the term $\mathsf{Adv}_{\texttt{one-way}}$ in the second bound), or has to do an on-line dictionary attack, which only enables him to test one password per session (the term $q_s \times 2^{-m}$ in both bounds).

**PAKE security.** If we consider the previous model with the trivial (insecure) password hashing scheme, where salts are $\perp$ and $\mathsf{PHash}(\mathsf{param}, \perp, \pi) = \pi$, we get back to the real-or-random variant of the BPR security model [AFP05]. This stronger model seems to be verified by the GL construction.

### 4.2 Related-Password Model

Unfortunately, this model is not easy to use because it requires to take care of computation time of everything. In addition, this model is weak, since it does not take into account related passwords. We

thus introduce a stronger security model for VPAKE protocols, called *related-password model*, which solves the above issues.

This model directly yields a stronger model for PAKE which has the same advantages regarding the choice of the password distribution. In addition, the GL constructions and variants seem still secure in this stronger model, which means in particular that these constructions are secure even if the adversary chooses related passwords.

**Model.** Users are defined as above, with clients $C \in \mathcal{C}$, each with a unique password $\pi_C$, and servers $S \in \mathcal{S}$, each with a unique pair $(s_S, H_S)$ of salt and hash value of a password $\pi_S$, but they are not paired. Instead, the passwords $(\{\pi_C\}, \{\pi_S\})$ are drawn from a joint (polynomial-time samplable) distribution $\mathcal{D}'$. The security has to hold for any $\mathcal{D}'$.

Passwords can now be related and servers and clients need not to be paired beforehand, and in particular `Execute` and `Send` queries can be performed between any client and any server. We therefore need to slightly change the way `Test` queries are answered. We say that a server $S$ and a client $C$ are *compatible* if $\mathsf{PFullHash}(\mathsf{param}, s_S, \pi_C) = H_S$. Then if a `Test` query is done for a user instance $U^i$ which is partnered with some user instance $U'^j$, for which a `Test` query has already been done, and if $U$ and $U'$ are compatible, then we return the same session key as for $U'^j$. We remark that, with high probability, $S$ and $C$ are compatible if and only if $\pi_C = \pi_S$, thanks to second pre-image resistance, the pre-hash entropy preservation and the fact that $\mathcal{D}'$ is polynomial-time samplable. We also remark that this slight modification makes the model stronger by ensuring that if two incompatible users execute the protocol, they end up with independent random keys.

The main difference between the related-password model and the BPR-like model is not these above minor changes but actually concerns the advantage. The advantage is no more bounded by some fixed expression as before. Instead, it is bounded by the advantage of the adversary in another game in which he cannot win with non-negligible probability[3]. More precisely, we suppose there exists a polynomial time extractor $\mathsf{Extract}$ which takes as input a bit $B$ and a transcript between a client and a server and which outputs a salt $s$ and a hash value $H$ if $B = 0$ and a pre-hash value $P$ if $B = 1$. Intuitively, if $B = 0$, $(s, H)$ is a salt and a hash value compatible with the password assigned to the server, while, if $B = 1$, $P$ is the pre-hash value corresponding to the password used by the client.

Let us call $\mathbf{G}_{\mathtt{real}}$ the game defined as in the BPR-like model, with the above changes, and let $\mathbf{G}_{\mathtt{ideal}}$ be a new game where every query to the `Execute` and `Send` oracles are answered using a fixed dummy (i.e., invalid in real life) password denoted by 0, and all the non-trivial `Test` queries are answered with a random session key (while the trivial cases are still answered by $\perp$). In addition, at the end of $\mathbf{G}_{\mathtt{ideal}}$, for each user instance $U^i$ involved in an active session (with no partner, but with the adversary) for which a non-trivial `Test` query has been asked, we call the $\mathsf{Extract}$ procedure for the transcript seen by $U^i$ and get either a hash value $H$ with a salt $s$ if $U$ is a client (with password $\pi = \pi_U$ and so pre-hash value $P = \mathsf{PPreHash}(\mathsf{param}, \pi_U)$) or a pre-hash value $P$ if $U$ is a server (with salt $s = s_S$ and hash value $H = H_S = \mathsf{PFullHash}(\mathsf{param}, \pi_S)$). If the pair $(s, H)$ corresponds to the pre-hash value $P$ (i.e., if $H = \mathsf{PHash}(\mathsf{param}, s, P)$), then we directly return the correct bit $b$, instead of returning the bit $b'$ given by the adversary. In other word, in the game $\mathbf{G}_{\mathtt{ideal}}$, we make the adversary win if we are able to extract a valid credential from the transcript for which he asked a non-trivial `Test` query.

Then, a password hashing scheme is secure in the related-password model if, for all polynomial time adversaries $\mathcal{A}$:

$$\mathsf{Adv}_{\mathtt{real}}(\mathcal{A}, \mathfrak{K}) \leq \mathsf{Adv}_{\mathtt{ideal}}(\mathcal{A}, \mathfrak{K}) + \mathrm{negl}(\mathfrak{K}),$$

[3] At least if passwords are chosen as in the BPR version. If passwords are chosen according to a more general joint distribution $\mathcal{D}'$, then the adversary may win. But, intuitively the advantage of the adversary will only be due to "cheating" correlations or "trapdoors" in $\mathcal{D}'$ and not due to weaknesses in the protocol.

where $\mathsf{Adv}_{\mathtt{real}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}_{\mathtt{ideal}}(\mathcal{A}, \mathfrak{K})$ are the advantage of a polynomial time adversary $\mathcal{A}$ in $\mathbf{G}_{\mathtt{real}}$ and $\mathbf{G}_{\mathtt{ideal}}$ respectively.

In Appendix C.3, we show that a VPAKE protocol secure in the related-password model is also secure in the BPR-like model.

# 5 VPAKE Constructions

## 5.1 Tools

**Smooth Projective Hash Functions.** Projective hash function families were first introduced by Cramer and Shoup [CS02]. Here we use the formalization from [BBC+13b]: Let $X$ be the domain of these functions and let $L$ be a certain subset of this domain (a language). A key property of these functions is that, for words $c$ in $L$, their values can be computed by using either a *secret* hashing key $\mathsf{hk}$ or a *public* projection key $\mathsf{hp}$ but with a witness $w$ of the fact that $c$ is indeed in $L$:
- $\mathsf{HashKG}(L)$ generates a hashing key $\mathsf{hk}$ for the language $L$;
- $\mathsf{ProjKG}(\mathsf{hk}, L, c)$ derives the projection key $\mathsf{hp}$, possibly depending on the word $c$;
- $\mathsf{Hash}(\mathsf{hk}, L, c)$ outputs the hash value from the hashing key, for any word $c \in X$;
- $\mathsf{ProjHash}(\mathsf{hp}, L, c, w)$ outputs the hash value from the projection key $\mathsf{hp}$, and the witness $w$, for a word $c \in L$.

On the one hand, the *correctness* of the SPHF assures that if $c \in L$ with $w$ a witness of this fact, then $\mathsf{Hash}(\mathsf{hk}, L, c) = \mathsf{ProjHash}(\mathsf{hp}, L, c, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $c \notin L$, $\mathsf{Hash}(\mathsf{hk}, L, c)$ is *statistically* indistinguishable from a random element, even knowing $\mathsf{hp}$.

Note that HashKG and ProjKG can just depend partially on $L$ (i.e., can only depend on a superset $L'$), and not at all on $c$: we then note $\mathsf{HashKG}(L')$ and $\mathsf{ProjKG}(\mathsf{hk}, L', c)$. In addition, if HashKG and ProjKG do not depend on $c$ and verify a slightly stronger smoothness property (called adaptive smoothness, which holds even if $c$ is chosen after $\mathsf{hp}$), we say the SPHF is a KVSPHF. Otherwise, it is said to be a GLSPHF. See [BBC+13b] for details on GLSPHF and KVSPHF and language definitions.

**Encryption Scheme.** A labeled public-key encryption scheme is defined by three algorithms:
- $\mathsf{KG}(1^{\mathfrak{K}})$ generates a key pair: a public key $\mathsf{pk}$ and a secret key $\mathsf{sk}$;
- $\mathsf{Enc}^{\ell}(\mathsf{pk}, x; r)$ encrypts the message $x$ under the key $\mathsf{pk}$ and the label $\ell$, and using the random coins $r$;
- $\mathsf{Dec}^{\ell}(\mathsf{sk}, c)$ decrypts the ciphertext $c$ with the label $\ell$ using the secret key $\mathsf{sk}$.

In this paper, we suppose that encryption schemes are `IND-CPA` or `IND-CCA` [BDPR98].

## 5.2 Intuitive Framework

**Intuitive Framework.** Basically if we restrict ourselves to password hashing schemes, our constructions work as follows. When a client $C$ wants to authenticate to a server $S$, it sends a commitment of the pre-hash value $P_C = \mathsf{PPreHash}(\mathsf{param}, \pi_C)$ of his password $\pi_C$ while the server sends a commitment of the hash value $H_S$ corresponding to the client's password, together with the salt $s_S$ used for computing this hash value. Using this salt $s_S$, the client can compute the hash value $H_C = \mathsf{PHash}(\mathsf{param}, s_S, P_C)$. If the client and the server are compatible and honest, $H_C = H_S$. Then SPHFs are used to ensure that the server committed to $H_S = H_C$ and the client committed to a pre-hash value $P_C$ verifying $\mathsf{PHash}(\mathsf{param}, s_S, P_C) = H_S$.

**Link with LAKE.** Our construction can roughly be seen as a particular case of a LAKE [BBC+13a], where the private information is the hash value $H_S = H_C$, and where the server does not hold any word and the client holds a word $P_C = \mathsf{PPreHash}(\mathsf{param}, \pi_C) \in \{P \mid \mathsf{PHash}(\mathsf{param}, s_S, P) = H_S\}$.

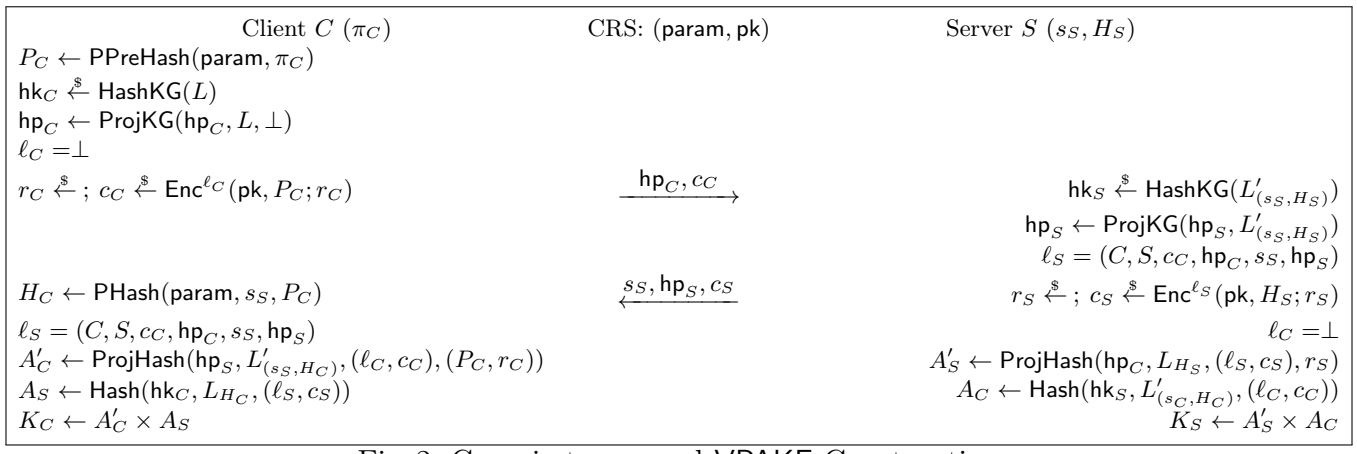


Fig. 2: Generic two-round VPAKE Construction

However, if we actually base our construction on existing LAKE constructions such as [BBC+13a, BBC+13b, ABB+13], we cannot directly compose these LAKEs with a password hashing scheme, because these constructions have not be proven under strong enough models for our purpose. More precisely, the original UC model of LAKE in [BBC+13a] does not support corruption (of the server private information as in our BPR-like model), while the BPR-like model of LAKE in [BBC+13b] does not compose easily. In addition, in our case, we have to take care of the fact that a malicious server can send a malicious salt $s_S$, which was not taken into accound in these previous models[4].

However, it seems that using any of the constructions mentionned above should work. Since in this paper, we are not interested in UC-secure schemes, our generic VPAKE scheme in Section 5.3 is based on a two-round variant of the LAKE construction in [BBC+13b], using a KVSPHF and a GLSPHF instead of two KVSPHFs. Replacing a KVSPHF by a GLSPHF often makes the protocol more efficient by reducing the amount of data exchanged, but also makes the protocol two-round instead of one-round. We also provide a one-round version of our protocol later in the paper.

### 5.3 Generic Two-Round Construction

**Construction.** Let us consider an IND-CCA encryption scheme $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$, a password hashing scheme $\mathsf{PH} = (\mathsf{Setup}, \mathsf{PSalt}, \mathsf{PPreHash}, \mathsf{PHash})$ and let us suppose we have a KVSPHF and a GLSPHF (respectively) for the two following families of languages:

$$L_H = \{(\ell, c) \mid \exists r,\ c = \mathsf{Enc}^{\ell}(\mathsf{pk}, H; r)\}$$
$$L'_{(s,H)} = \{(\ell, c) \mid \exists P, \exists r,\ c = \mathsf{Enc}^{\ell}(\mathsf{pk}, P; r) \text{ and } H = \mathsf{PHash}(\mathsf{param}, s, P)\},$$

with param and pk global parameters in the common reference string CRS, generated as follows: $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}, 1^n)$ and $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KG}(1^{\mathfrak{K}})$. We also suppose the HashKG and ProjKG for $L_H$ does not depend on $H$, or in other words, just works for $L = \{(\ell, c) \mid \exists r, \exists H,\ c = \mathsf{Enc}^{\ell}(\mathsf{pk}, H; r)\} \supset L_H$ (for any $H$).

Then our two-round construction is depicted in Figure 2, where $\times$ is a commutative operation between hash values such that if $A$ is a uniform hash value and $B$ is any hash value, $A \times B$ is uniform (often hash values live in a group and $\times$ is just the group law).

**Security and Small Improvement.** This scheme is secure under the related-password model for VPAKE. The security proof is similar to the one in [BBC+13b] and can be found in Appendix C.4.

We remark that, in the proof, the IND-CCA security is only required to the encryption scheme used by the server. The client may use only an IND-CPA encryption scheme without label.

[4] In these previous models, there is a public information which can be used as a salt, but it is supposed that both users agree on it, while in our case, the server chooses it. The client has even no way to verify that the server always use the same salt, between two executions of the protocol, since the client only knows the user's password.

**One-Round Variant.** If both SPHFs are KVSPHFs, then the previous protocol is actually one-round (after a slight modification on labels[5]): the two flows can be sent simultaneously. The proof of security is a slight variant of the previous proof, using the same ideas as in [KV11]. However, in this case, both encryptions schemes have to be IND-CCA.

### 5.4 Instantiations

In this section we briefly show how to instantiate the generic scheme with the two password hashing schemes described in Sections 3.3 and 3.2.

To instantiate our generic scheme, we just need to choose an IND-CCA encryption scheme and to construct SPHFs for the languages $L_H$ and $L'_{(s,H)}$. Both our instantiations use a labeled Cramer-Shoup encryption on vectors recalled in Appendix B.1, which can also be used to encrypt bit strings (such as $\mathcal{H}(\pi)$ for the password hashing based on random oracles), while SPHFs can easily be constructed using the generic framework introduced in [BBC+13b]. For the first instantiation, we propose two variants: one in two rounds and one in one round but slightly less efficient. For the second instantiation, we only propose it as a two-round protocol. Details can be found in Appendix B, due to lack of space.

The security of the instantiations directly comes from the security of our password hashing schemes (proven in the generic group model and in the generic multi-linear group model) and of the security of the generic scheme. We remark that if we assume the security of our password hashing schemes or if we prove them without any idealized models, which could be done under non-standard assumption, our instantiations would be secure in the standard model.

## Acknowledgments

The first author would like to thank Hang Zhou for proof-reading a preliminary version of the paper. This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet.

## References

ABB+13. M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, and D. Pointcheval. Sphf-friendly non-interactive commitments. Cryptology ePrint Archive, Report 2013/588, 2013. http://eprint.iacr.org/. (Pages 2 and 12.)

ACP09. M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *CRYPTO 2009*, *LNCS* 5677, pages 671–689. Springer, August 2009. (Page 2.)

AFP05. M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In *PKC 2005*, *LNCS* 3386, pages 65–84. Springer, January 2005. (Pages 2, 8, and 9.)

AP05. M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In *CT-RSA 2005*, *LNCS* 3376, pages 191–208. Springer, February 2005. (Page 2.)

BBC+13a. F. Ben Hamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In *PKC 2013*, *LNCS* 7778, pages 272–291. Springer, February / March 2013. (Pages 3, 11, and 12.)

BBC+13b. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 449–475. Springer, August 2013. (Pages 2, 3, 11, 12, 13, 14, 16, 18, and 19.)

BBC+13c. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New smooth projective hash functions and one-round authenticated key exchange. Cryptology ePrint Archive, Report 2013/034, 2013. http://eprint.iacr.org/. Full version of the SPHF part of [BBC+13b]. (Pages 3 and 18.)

BBS03. M. Bellare, A. Boldyreva, and J. Staddon. Randomness re-use in multi-recipient encryption schemeas. In *PKC 2003*, *LNCS* 2567, pages 85–99. Springer, January 2003. (Page 16.)

BCP03. E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In *ACM CCS 03*, pages 241–250. ACM Press, October 2003. (Page 2.)

BCP04. E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. In *PKC 2004*, *LNCS* 2947, pages 145–158. Springer, March 2004. (Page 2.)

[5] Namely, $\ell_C = (C, S, \mathsf{hp}_C)$ and $\ell_S = (C, S, \mathsf{hp}_S)$.

BDJR97. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, October 1997. (Page 2.)

BDPR98. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, *LNCS* 1462, pages 26–45. Springer, August 1998. (Page 11.)

BM92. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. (Pages 1 and 2.)

BM93. S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM CCS 93*, pages 244–250. ACM Press, November 1993. (Page 1.)

BPR00. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, *LNCS* 1807, pages 139–155. Springer, May 2000. (Pages 2, 8, and 9.)

BR13. Z. Brakerski and G. N. Rothblum. Obfuscating conjunctions. In *CRYPTO 2013, Part II*, *LNCS* 8043, pages 416–434. Springer, August 2013. (Pages 3, 7, and 16.)

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. (Page 2.)

CHK+05. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In *EUROCRYPT 2005*, *LNCS* 3494, pages 404–421. Springer, May 2005. (Page 2.)

CLT13. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 476–493. Springer, August 2013. (Pages 3 and 16.)

CS02. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, *LNCS* 2332, pages 45–64. Springer, April / May 2002. (Pages 2 and 11.)

DH76. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Page 1.)

GGH13. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT 2013*, *LNCS* 7881, pages 1–17. Springer, May 2013. (Pages 3, 14, and 16.)

GL03. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT 2003*, *LNCS* 2656, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`. (Pages 2 and 26.)

GMR06. C. Gentry, P. MacKenzie, and Z. Ramzan. A method for making password-based key exchange resilient to server compromise. In *CRYPTO 2006*, *LNCS* 4117, pages 142–159. Springer, August 2006. (Page 2.)

KOY01. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT 2001*, *LNCS* 2045, pages 475–494. Springer, May 2001. (Page 2.)

KV11. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC 2011*, *LNCS* 6597, pages 293–310. Springer, March 2011. (Pages 2 and 13.)

Pol00. J. M. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13(4):437–447, 2000. (Page 7.)

Sch01. C.-P. Schnorr. Small generic hardcore subsets for the discrete logarithm: Short secret dl-keys. *Inf. Process. Lett.*, 79(2):93–98, 2001. (Pages 6 and 20.)

Sha71. D. Shanks. Class number, a theory of factorization, and genera. In *1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969)*, pages 415–440. AMS, Providence, R.I., 1971. (Page 7.)

# A Graded Rings

## A.1 Graded Rings

Let us first recall the notion of graded ring introduced in [BBC+13b], restricted to the "asymmetrical" case. Such graded rings are a generalization of asymmetric bilinear groups and can be used as a practical abstraction of asymmetrical multi-linear maps coming from the framework of Garg, Gentry and Halevi [GGH13].

**Indexes Set.** Let us consider a finite set of indexes[6] $\Lambda = \mathbb{Z}_2^\tau \subset \mathbb{N}^\tau$. In addition to considering the addition law + over $\Lambda$, we also consider $\Lambda$ as a bounded lattice, with the two following laws:

$$\sup(\boldsymbol{v}, \boldsymbol{v}') = (\max(\boldsymbol{v}_1, \boldsymbol{v}'_1), \ldots, \max(\boldsymbol{v}_\tau, \boldsymbol{v}'_\tau)) \qquad \inf(\boldsymbol{v}, \boldsymbol{v}') = (\min(\boldsymbol{v}_1, \boldsymbol{v}'_1), \ldots, \min(\boldsymbol{v}_\tau, \boldsymbol{v}'_\tau)).$$

We also write $\boldsymbol{v} < \boldsymbol{v}'$ (resp. $\boldsymbol{v} \leq \boldsymbol{v}'$) if and only if for all $i \in \{1, \ldots, \tau\}$, $\boldsymbol{v}_i < \boldsymbol{v}'_i$ (resp. $\boldsymbol{v}_i \leq \boldsymbol{v}'_i$). Let $\bar{0} = (0, \ldots, 0)$ and $\top = (1, \ldots, 1)$, be the minimal and maximal elements. Finally, let $\boldsymbol{e}_i$ be the $i$-th vector of the canonical base of $\Lambda$ ($\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$).

[6] In the original definition in [BBC+13b], the authors consider more general sets $\Lambda$.

**Graded Ring.** The $\tau$-graded ring over a commutative ring $R$ is the set $\mathfrak{G} = \Lambda \times R = \{[\boldsymbol{v}, x] \mid \boldsymbol{v} \in \Lambda, x \in R\}$, where $\Lambda = \mathbb{Z}_2^\tau$, with two binary operations $(\oplus, \odot)$ defined as follows:
- for every $u_1 = [\boldsymbol{v}_1, x_1], u_2 = [\boldsymbol{v}_2, x_2] \in \mathfrak{G}$: $u_1 \oplus u_2 \coloneqq [\sup(\boldsymbol{v}_1, \boldsymbol{v}_2), x_1 + x_2]$;
- for every $u_1 = [\boldsymbol{v}_1, x_1], u_2 = [\boldsymbol{v}_2, x_2] \in \mathfrak{G}$: $u_1 \odot u_2 \coloneqq [\boldsymbol{v}_1 + \boldsymbol{v}_2, x_1 \cdot x_2]$ if $\boldsymbol{v}_1 + \boldsymbol{v}_2 \in \Lambda$, or $\perp$ otherwise, where $\perp$ means the operation is undefined and cannot be done.

We remark that $\odot$ is only a partial binary operation and we use the following convention: $\perp \oplus u = u \oplus \perp = u \odot \perp = \perp \odot u = \perp$, for any $u \in \mathfrak{G} \cup \{\perp\}$. Let also $\mathfrak{G}_{\boldsymbol{v}}$ be the additive group $\{u = [\boldsymbol{v}', x] \in \mathfrak{G} \mid \boldsymbol{v}' = \boldsymbol{v}\}$ of graded ring elements of index $\boldsymbol{v}$. We will make natural use of vector and matrix operations over graded ring elements.

### A.2 Cyclic Groups and Asymmetric Bilinear Groups

Let us now show that cyclic groups and asymmetric bilinear groups of order $p$ can be seen as graded rings over $R = \mathbb{Z}_p$:

**Cyclic groups:** $\tau = 1$. More precisely, elements $[0, x]$ of index 0 correspond to scalars $x \in \mathbb{Z}_p$ and elements $[1, x]$ of index 1 correspond to group elements $g^x \in \mathbb{G}$.

**Asymmetric bilinear groups** $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$: $\tau = 2$. More precisely, we can consider the following map: $[(0,0), x]$ corresponds to $x \in \mathbb{Z}_p$, $[\boldsymbol{e}_1, x]$ corresponds to $g_1^x \in \mathbb{G}_1$, $[\boldsymbol{e}_2, x]$ corresponds to $g_2^x \in \mathbb{G}_2$ and $[(1,1), x]$ corresponds to $e(g_1, g_2)^x \in \mathbb{G}_T$.

**Notations.** We have chosen an additive notation for the group law in $\mathfrak{G}_{\boldsymbol{v}}$. On the one hand, this a lot easier to write down generic things, but, on the other hand, it is a bit cumbersome for bilinear groups to use additive notations. When we provide an example with a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, we use multiplicative notation $\cdot$ for the law in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and additive notation $+$ for the law in $\mathbb{Z}_p$, as soon as it is not too complicated. Therefore, for any $x, y \in \mathbb{Z}_p$, $u_1, v_1 \in \mathbb{G}_1$, $u_2, v_2 \in \mathbb{G}_2$ and $u_T, v_T \in \mathbb{G}_T$, we have:

$$
\begin{array}{lll}
x \oplus y = x + y & & x \odot y = x \cdot y = xy \\
u_1 \oplus v_1 = u_1 \cdot v_1 = u_1 v_1 & & x \odot u_1 = u_1^x \\
u_2 \oplus v_2 = u_2 \cdot v_2 = u_2 v_2 & & x \odot u_1 = u_1^x \\
u_T \oplus v_T = u_T \cdot v_T & u_1 \odot u_2 = e(u_1, u_2) & x \odot u_T = u_T^x.
\end{array}
$$

### A.3 Assumptions

Let us recall the DDH (in cyclic groups) and the SXDH (in asymmetric bilinear groups) assumptions before introducing the GXDH assumption, which can be seen as an extension of these previous assumptions to graded rings.

**Definition 1 (Decisional Diffie-Hellman Assumption (DDH)).** *Let $\mathbb{G}$ be a multiplicative cyclic group of order $p$, and let $g$ be a generator of $\mathbb{G}$. The DDH assumption says that, when we are given $(g^a, g^b, g^c) \in \mathbb{G}^3$, with $a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = ab$ (a DH tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple).*

**Definition 2 (Symmetric External DDH Assumption (SXDH)).** *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be an asymmetric bilinear group. The SXDH assumption says that the DDH is hard in $\mathbb{G}_1$ and in $\mathbb{G}_2$.*

**Definition 3 (Graded External DDH Assumption (GXDH)).** *Let $\mathfrak{G}$ be a $\tau$-graded ring. The GXDH assumption says that the DDH is hard in each $\mathfrak{G}_{\boldsymbol{e}_i}$ (for $i = 1, \ldots, \tau$). In other words, for $i = 1, \ldots, \tau$, when we are given $(g, a \odot g, b \odot g, c \odot g)$ with $g$ a generator of $\mathfrak{G}_{\boldsymbol{e}_i}$ and $a, b \xleftarrow{\$} R$, it is hard to decide whether $c = ab$ (a DH tuple) or $c \xleftarrow{\$} R$ (a random tuple).*

### A.4 Candidate Instantiations of Graded Rings with $\tau > 2$

We do not know any exact construction of graded rings for $\tau > 2$. However, Garg, Gentry and Halevi proposed a generalization of graded rings[7] in a seminal paper [GGH13], called graded encoding schemes, where each element may have multiple encodings. In addition, they proposed an approximate instantiation of graded encoding schemes based on ideal lattices, where encodings are noisy and noise increases at each operations.

Unfortunately their construction does not suit our purpose because the GXDH assumption does not hold. That is why we use the construction of Coron, Lepoint and Tibouchi [CLT13] over the integers instead (denoted CLT). Let us now list all differences between ideal graded rings and the CLT construction, and show how to adapt our schemes and proofs.

**Multiple Encodings.** The first difference is the fact that in CLT, each element has multiple encodings. This is actually not a problem at all in our case, since the CLT construction also provides a way to test the equality of two elements and a way to extract a canonical representation of any element (such that the canonical representation of a random element is random).

**Noise.** In CLT, encodings are noisy, and noise increases after each operation. Parameters have to be chosen accordingly and the initial noise has to be small enough, as already explained in [BR13] for example.

**Base Ring $R \neq \mathbb{Z}_p$ and Sampling.** Contrary to what we suppose in the core of the paper, the base ring $R$ is not $\mathbb{Z}_p$ but a direct product $\mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ with $g_1, \ldots, g_n$ distinct unknown primes of size at least $2\mathfrak{K}$. In addition, it is only possible to sample random elements in $R$; using some specific element and inverting elements is not possible. This is not a problem for our constructions, since we always sample random elements from $\mathbb{Z}_p$, and we could replace $\mathbb{Z}_p$ by $R$.

It remains to show that the security still holds when elements are chosen in $R$ instead of $\mathbb{Z}_p$ and with the above restrictions. This is slightly more involved and is not done very thoroughly. But the main idea is to remark that a random element $x$ in $R$ is such that $x \mod g_i$ is a generator of $\mathbb{Z}_{g_i}$ for all $i$, with high probability.

## B Instantiations

In this appendix, we show the details of the two instantiations of our generic two-round VPAKE from Section 5.4, after recalling the labeled Cramer-Shoup encryption scheme of vectors and of bit-strings, which is the encryption scheme used in our VPAKE instantiations.

### B.1 Labeled Cramer-Shoup Encryption of Vectors

**Encryption of Vectors.** For our VPAKE constructions, we use a variant of the Cramer-Shoup encryption scheme for vectors of $m$ messages, with randomness reuse [BBS03]. This scheme has already been used in [BBC+13b]. Let $\mathbb{G}$ be a cyclic group of order $p$, with two independent generators $g$ and $h$. The secret decryption key is a random vector $\mathsf{sk} = (x_1, x_2, y_1, y_2, z_1, \ldots, z_m) \xleftarrow{\$} \mathbb{Z}_p^{4+m}$ and the public encryption key is $\mathsf{pk} = (g, h, c = g^{x_1}h^{x_2}, d = g^{y_1}h^{y_2}, f_1 = g^{z_1}, \ldots, f_m = g^{z_m}, H_{\mathsf{CS}})$, where $H_{\mathsf{CS}}$ is randomly chosen in a collision-resistant hash function family $\mathcal{H}_{\mathsf{CS}}$ (actually, second-preimage resistance is enough). For a message-vector $\boldsymbol{M} = (M_i)_{i=1,\ldots,m} \in \mathbb{G}^m$, the multi-Cramer-Shoup encryption is defined as $m\text{-}\mathsf{MCS}^{\ell}_{\mathsf{pk}}(\boldsymbol{M}; r) = (u = g^r, v = h^r, e_1 = f_1^r \cdot M_1, \ldots, e_m = f_m^r \cdot M_m, w = (cd^{\theta})^r)$, where $\theta = H_{\mathsf{CS}}(\ell, u, v, e_1, \ldots, e_m)$. Such a ciphertext $C = (u, v, e_1, \ldots, e_m, w)$ is

[7] Actually graded rings were inspired from their graded encoding scheme.

decrypted by $M_i = e_i/u^{z_i}$, after having checked the validity of the ciphertext, $w \stackrel{?}{=} u^{x_1+\theta y_1} v^{x_2+\theta y_2}$. This multi-Cramer-Shoup encryption scheme, denoted $\mathsf{MCS}$, is $\mathtt{IND\text{-}CCA}$ under the $\mathsf{DDH}$ assumption.

We remark that a public key for the labeled Cramer-Shoup scheme for vectors of $m$ messages can also be used to encrypt fewer messages (by ignoring the useless $f_i$'s).

**Encryption of Bit-Strings.** The above encryption scheme can be used to encrypt bit-strings $x \in \{0,1\}^{|p|}$, just by encrypting the vector $(g^{x[1]}, \ldots, g^{x[|p|]})$.

**Extension to Asymmetric Graded Rings.** The above encryption schemes can be trivially used in asymmetric graded rings, by replacing $\mathbb{G}$ by $\mathfrak{G}_{\boldsymbol{v}}$ for any index $\boldsymbol{v}$. The scheme is $\mathtt{IND\text{-}CCA}$ secure as long as the $\mathsf{DDH}$ holds in $\mathfrak{G}_{\boldsymbol{v}}$.

### B.2 Instantiation Based on Random Oracles

For this instantiation we propose two variants: one in two rounds in a cyclic group $\mathbb{G}_1$ and one in one round in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, e)$. The first one is very efficient, while the second one is slightly inefficient due to the use of complex $\mathsf{SPHF}$s for quadratic pairing equations and encryption in both $\mathbb{G}_1$ and $\mathbb{G}_2$ for $P$.

**Two-Round Instantiation.** For this construction let $\mathbb{G}_1$ be a cyclic group of order $p$. We suppose $p$ has size $|p| = 2\mathfrak{K}$. Let $\mathsf{pk}_1 = (g_1, h_1, c_1, d_1, f_{1,1}, \ldots, f_{1,|p|}, H_{\mathsf{CS}})$ be the public key of the labeled Cramer-Shoup encryption scheme for vectors of size $|p|$ in $\mathbb{G}_1$, and let $\mathsf{param} = \mathbb{G}_1$ be the parameters of the password hashing scheme in Section 3.2.

Using notations of the generic two-round scheme in Section 5.3, we then have:

$$\begin{aligned} c_S &= \mathsf{Enc}^{\ell_S}(\mathsf{pk}_1, H_S; r_S) = 1\text{-}\mathsf{MCS}^{\ell_S}_{\mathsf{pk}_1}((H_S); r_S) \\ &= (u_S = g_1^{r_S}, v_S = h_1^{r_S}, e_S = f_{1,1}^{r_S} \cdot H_S, w_S = (c_1 d_1^{\theta_S})^{r_S}), \end{aligned}$$

with $\theta_S = H_{\mathsf{CS}}(\ell_S, u_S, v_S, e_S)$, since $H_S = s_S^{P_S} \in \mathbb{G}_1$ is a group element ($\pi_S$ being the password used to generated the password hash of the server, $P_s = \mathsf{PPreHash}(\mathsf{param}, \pi_S)$ the pre-hash value of $\pi_S$, and $s_S \in \mathbb{G}_1$ the seed) and

$$\begin{aligned} c_C &= \mathsf{Enc}^{\ell_C}(\mathsf{pk}_1, P; r_{C,1}) = |p|\text{-}\mathsf{MCS}^{\ell_C}_{\mathsf{pk}_1}((g_1^{P[i]})_i; r_{C,1}) \\ &= (u_{C,1} = g_1^{r_{C,1}}, v_{C,1} = h_1^{r_{C,1}}, e_{C,1,1} = f_{1,1}^{r_{C,1}} \cdot g_1^{P[i]}, \ldots, e_{C,1,|p|} = f_{1,|p|}^{r_{C,1}} \cdot g_1^{P[|p|]}, w_C = (c_1 d_1^{\theta})^{r_{C,1}}) \end{aligned}$$

with $\theta_C = H_{\mathsf{CS}}(\ell_C, u_{C,1}, v_{C,1}, (e_{C,1,i})_i)$ and where $P = \mathcal{H}(\pi)$ is a scalar in $\mathbb{Z}_p$, and so can be considered as a $|p|$ bit-string.

It remains to construct a $\mathsf{KVSPHF}$ for

$$\begin{aligned} L_H &= \{(\ell, c) \mid \exists r,\ \mathsf{Enc}^{\ell}(\mathsf{pk}_1, H; r)\} \\ &= \{(\ell, c = (u, v, e, w)) \mid \exists r \in \mathbb{G},\ u = g_1^r, v = h_1^r, e = f_{1,1}^r \cdot H \text{ and } w = (c_1 d_1^{\theta})^r\} \end{aligned}$$

with $\theta = H_{\mathsf{CS}}(\ell, u, v, e)$ and a $\mathsf{GLSPHF}$ or a $\mathsf{KVSPHF}$ for

$$\begin{aligned} L'_{(s,H)} &= \{(\ell, c) \mid \exists P, r,\ \mathsf{Enc}^{\ell}(\mathsf{pk}_1, P; r) = c \text{ and } \mathsf{PHash}(\mathsf{param}, s, P) = H\} \\ &= \left\{ (\ell, c = (u, v, (e_i)_i, w)) \;\middle|\; \begin{array}{l} \exists P \in \{0,1\}^{\mathfrak{K}},\ \exists r \in \mathbb{G}, \\ u = g_1^r, v = h_1^r, e_1 = f_{1,1}^r \cdot g_1^{P[0]}, \ldots, e_{|p|} = f_{1,|p|}^r \cdot g_1^{P[|p|-1]}, \\ w = (c_1 d_1^{\theta})^r, H = s^P \end{array} \right\}, \end{aligned}$$

with the correct $\theta = H_{\mathsf{CS}}(\ell, u, v, (e_i)_i)$.

*KVSPHF for $L_H$.* $L_H$ is just the language of Cramer-Shoup ciphertext of some given value $H$. In [BBC+13b], the authors already introduced a KVSPHF for this exact language. Let us just recall how to write this SPHF in the generic framework introduced in [BBC+13b], as a warm-up.

We use the following matrices:

$$\Gamma = \begin{pmatrix} g_1 & 1 & h_1 & f_{1,1} & c_1 \\ 1 & g_1 & 1 & 1 & d_1 \end{pmatrix} \qquad \begin{aligned} \boldsymbol{\lambda} &= (r, r\theta) \\ \boldsymbol{\lambda} \cdot \Gamma &= (g_1^{r+r\theta}, h_1^r, f_{1,1}^r, (c_1 d_1^\theta)^r) \\ \Theta(C) &= (u, v, e/H, w) \end{aligned}$$

With $\mathsf{hk} = (\eta_1, \eta_2, \alpha, \beta, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$, we get $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} h_1^\alpha f_{1,1}^\beta c_1^\mu, \mathsf{hp}_2 = g_1^{\eta_2} d_1^\mu) \in \mathbb{G}_1^2$.

We remark that this KVSPHF does not need to use $H$ in HashKG nor in ProjKG, which is required for the generic scheme.

*GLSPHF for $L'_{(s,H)}$.* This language is very similar to the one used for blind signatures in [BBC+13b], except we use Cramer-Shoup ciphertexts instead of ElGamal ones.

**One-Round Instantiation.** For this construction, let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear group of order $p$, with $g_1$ and $g_2$ two group generators. We also add to the common reference string a generator $g_2$ of $\mathbb{G}_2$.

The main difficulty is to construct a KVSPHF for $L'_{(s,H)}$. We do not know to construct such a KVSPHF for the current language, so we slightly change the way $P$ is encrypted. Instead of encrypting it only in $\mathbb{G}_1$, with also encrypt it in $\mathbb{G}_2$. More precisely, we consider a second public key $\mathsf{pk}_2 = (g_2, h_2, c_2, d_2, f_{2,1}, \ldots, f_{2,|p|})$ for the Cramer-Shoup encryption scheme for vectors in $\mathbb{G}_2$. Then we set:

$$\begin{aligned} c_C &= \mathsf{Enc}^{\ell_C}((\mathsf{pk}_1, \mathsf{pk}_2), \pi_C; (r_{C,1}, r_{C,2})) \\ &= (u_{C,1} = g_1^{r_{C,1}}, v_{C,1} = h_1^{r_{C,1}}, e_{C,1,1} = f_{1,1}^{r_{C,1}} \cdot g_1^{P[i]}, \ldots, e_{C,1,|p|}^{r_{C,1}} = f_{1,|p|} \cdot g_1^{P[|p|]}, w_C = (c_1 d_1^\theta)^{r_C}, \\ &\quad u_{C,2} = g_2^{r_{C,2}}, v_{C,2} = h_2^{r_{C,2}}, e_{C,2,1} = f_{2,1}^{r_{C,2}} \cdot g_2^{P[i]}, \ldots, e_{C,2,|p|}^{r_{C,2}} = f_{2,|p|} \cdot g_2^{P[|p|]}, w_C = (c_2 d_2^\theta)^{r_{C,2}}) \end{aligned}$$

where $\theta = H_{\mathsf{CS}}(\ell_C, u_{C,1}, v_{C,1}, (e_{C,1,i})_i, u_{C,2}, v_{C,2}, (e_{C,2,i})_i)$ is the same for the encryption in $\mathbb{G}_1$ and $\mathbb{G}_2$ to ensure non-malleability (see appendices of the full versions of [BBC+13b], for details on this trick).

It remains to construct the KVSPHF for the associated language $L'_{(s,H)}$. For this, we just remark that some word $(\ell, c)$ is in this language if $c$ is a valid ciphertext under label $\ell$ of a vector $(X_{1,1} = g_1^{x_{1,1}}, \ldots, X_{1,|p|} = g_1^{x_{1,|p|}}, X_{2,1} = g_2^{x_{2,1}}, \ldots, X_{2,|p|} = g_2^{x_{2,|p|}}) \in \mathbb{G}_1^{|p|} \times \mathbb{G}_2^{|p|}$ such that:

- $x_{1,i} = x_{2,i}$ and $x_{1,i} = 0$ or $1$, which is equivalent to say

$$e(X_{1,i}, g_2) = e(g_1, X_{2,i}) \quad \text{and} \quad e(X_{1,i}, X_{2,i}) = e(X_{1,i}, g_2),$$

  since the first equation ensures $x_{1,i} = x_{2,i}$ and the second equation ensures $x_{1,i}^2 = x_{1,i} x_{2,i} = x_{1,i}$, and so that either $x_{1,i} = 0$ or $x_{1,i} = 1$; and,
- $s^{x_{1,1} + 2x_{1,2} + \cdots + 2^{|p|} x_{1,|p|}} = H$.

For the first point, we can use the KVSPHF for systems of quadratic pairing equations over ciphertexts (in the last appendix of [BBC+13c]), and for the second the KVSPHF for systems of linear multi-exponentiation in [BBC+13b]. To mix up both SPHFs, we just need to take the product of both hash values (though, this could actually be improved a bit).

### B.3 Instantiation Based on Multi-Linear Maps

For this instantiation, we only have a two-round instantiation. Current construction methods seem to generate an exponential blow-up in $n$ for the size of a projection key hp for a KVSPHF for $L'_{s,H}$, except if the client encrypts some additionnal values (such as $a_{1,\pi[1]} \odot \cdots \odot a_{i,\pi[i]}$) but this is out of the scope of this article.

Let $\mathfrak{G}$ be a $(n+2)$-graded ring of order $p$, with $n$ the password length. Let $g_{\boldsymbol{e}_1}, \ldots, g_{\boldsymbol{e}_{n+2}}$ and $h_{n+2}$ be generators of $\mathfrak{G}_{\boldsymbol{e}_1}$, …, $\mathfrak{G}_{\boldsymbol{e}_{n+2}}$ and $\mathfrak{G}_{\boldsymbol{e}_{n+2}}$, respectively. We set $g_{\boldsymbol{v}} = \bigodot_{i \in \boldsymbol{v}} g_{\boldsymbol{e}_i}$ a generator of $\mathfrak{G}_{\boldsymbol{v}}$, and we write $g_0 = g_{\boldsymbol{e}_{n+2}}$ and $h_0 = g_{\boldsymbol{e}_{n+2}}$.

Let $\mathsf{pk}_0 = (g_0, h_0, c_0, d_0, f_1, \ldots, f_n, H_{\mathsf{CS}})$ be the public key of the labeled Cramer-Shoup encryption scheme in $\mathfrak{G}_{\boldsymbol{e}_{n+2}}$ for vectors of size $n$. This public key implicitely defines a public key for the labeled Cramer-Shoup encryption for elements in $\mathfrak{G}_\top$: $\mathsf{pk}_\top = (g_\top = g_0 \odot g', h_\top = h_0 \odot g', c_\top = c_0 \odot g', d_\top = d_0 \odot g', f_\top = f_1 \odot g', H_{\mathsf{CS}})$, with $g' = g_{\boldsymbol{v}}$ a generator of $\mathfrak{G}_{\boldsymbol{v}}$, with $\boldsymbol{v} = \top - \boldsymbol{e}_{n+2} = (1, \ldots, 1, 0)$. This implicitely derived public key is used by the server to encrypt $H_S$, while the initial public key is used by the client to encrypt its password $\pi_C$.

Let $\mathsf{param} = (\mathfrak{G}, (a_{i,b})_{i,b})$ be the parameters of the password hashing scheme in Section 3.2 (for this scheme the last dimension $e_{n+2}$ is just ignored, which does not change anything).

Using notations of the generic two-round scheme in Section 5.3, we then have:

$$\begin{aligned} c_S &= \mathsf{Enc}^{\ell_S}(\mathsf{pk}_0, H_S; r_S) = 1\text{-}\mathsf{MCS}^{\ell_S}_{\mathsf{pk}_0}((H_S); r_S) \\ &= (u_S = r_S \odot g_\top, v_S = r_S \odot h_\top, e_S = r_S \odot f_\top \oplus H_S \odot g_0, w_S = r_S \odot (c_\top \oplus (\theta_S \odot d_\top))), \end{aligned}$$

with $\theta_S = H_{\mathsf{CS}}(\ell_S, u_S, v_S, e_S)$ and:

$$\begin{aligned} c_C &= \mathsf{Enc}^{\ell_C}(\mathsf{pk}_1, \pi_C; r_C) = |p|\text{-}\mathsf{MCS}^{\ell_C}_{\mathsf{pk}_1}((\pi_C[i] \odot g_0)_i; r_C) \\ &= (u_C = r_C \odot g_0, v_C = r_C \odot h_0, (e_{C,i} = r_C \odot f_{1,i} \oplus \pi_C[i] \odot g_0), r_C \odot (c_0 \oplus (\theta_C \odot d_0))), \end{aligned}$$

with $\theta_C = H_{\mathsf{CS}}(\ell_C, u_C, v_C, (e_{C,i})_i)$.

Let us now construct the SPHFs.

*KVSPHF for $L_H$.* This one is identical to the one for the two-round instantiation based on random oracles.

*GLSPHF for $L'_{(s,H)}$.* This SPHF can be separated in two parts:

- one for proving that the ciphertext $c_C$ is a valid Cramer-Shoup ciphertext under the label $\ell$ of a vector of group elements $X_1 = g_0^{x_1}, \ldots, X_n = g_0^{x_n}$, with $x_1, \ldots, x_n \in \{0, 1\}$,
- one for proving that if we set $x = x_1 + 2x_2 + \cdots + 2^{n-1}x_n$, then $H = \mathsf{PHash}(\mathsf{param}, s, x)$, or in other words, $H = a_{1,x_1} \odot \cdots \odot a_{n,x_n} \odot s$.

The first part can be handled as in the two-round instantiation based on random oracles, i.e., as in the SPHF for blind signature scheme in [BBC+13b]. Let us therefore focus on the second part.

For this part, we use the following matrices:

$$\Gamma = \begin{pmatrix} g_0 & \eta_1 & & & & & & & \\ & g_0 & \ominus u & \ominus \epsilon_2 & & & & & \\ & & g_0 & \eta_2 & & & & & \\ & & & g_0 & \ominus u & \ominus \epsilon_3 & & & \\ & & & & g_0 & \eta_3 & & & \\ & & & & & \ddots & & & \\ & & 0 & & & & g_0 & \ominus u & \ominus \epsilon_n - 1 \\ & & & & & & g_0 & \eta_n - 1 & \\ & & & & & & & g_0 & \ominus u \quad \ominus \epsilon_n \\ & & & & & & & & g_0 \quad \eta_n \end{pmatrix}$$

$$\begin{aligned} \Theta(c) &= (s \odot u,\ s \odot \epsilon_1,\ 0,\ 0,\ \ldots,\ 0,\ H \odot g_0) \\ \boldsymbol{\lambda} &= (r \odot s,\ s \odot a_{1,\pi[1]},\ r \odot s \odot a_{1,\pi[1]}, \\ &\quad s \odot a_{1,\pi[1]} \odot a_{2,\pi[2]},\ r \odot s \odot a_{1,\pi[1]} \odot a_{2,\pi[2]},\ \ldots, \\ &\quad s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-2,\pi[n-2]},\ r \odot s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-2,\pi[n-2]}, \\ &\quad s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-1,\pi[n-1]},\ r \odot s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-1,\pi[n-1]}), \end{aligned}$$

with $\epsilon_i = e_i \odot a_{i,1} \oplus (g_0 \ominus e_i) \odot a_{i,0}$ and $\eta_i = f_i \odot (a_{i,1} \ominus a_{i,0})$. To see why this works, we first remark that, due to the first part of the SPHF, if we write $u = r \odot g_0$, we are sure that, for all $i$, $e_i = r \odot f_i \oplus x_i \odot g_0$, with $x_i \in \{0, 1\}$, and so $\epsilon_i = r \odot \eta_i \oplus (x_i \odot a_{i,1} \oplus (1 - x_i) \odot a_{i,0})$, in other words, $\epsilon_i = r \odot \eta_i \oplus a_{i,x_i}$. So, the SPHF just checks that $s \odot a_{1,x_1} \odot \cdots \odot a_{n,x_n} = H$.

# C Proofs

## C.1 Security of our Password Hashing Scheme Based on Random Oracles

In this section, we prove the tight one-wayness of our password hashing based on random oracles.

**Case with only One Hash Query.** Let us first consider the case with only one query **Hash**. For that purpose, let us first prove the following theorem, which is an extension of Theorem 2 from [Sch01].

**Theorem 4.** *Let $\mathbb{G}$ be a cyclic group of order $p$. Let $\mathcal{D}_{\mathcal{H}}$ be the uniform distribution over the functions from $\{0,1\}^n$ to $\mathbb{Z}_p$ and let $\mathcal{D}$ be a polynomially samplable distribution over $\{0,1\}^n$ of min-entropy at least $\beta$. Then, if $p - 2^n > 2^{2n}\sqrt{p}$, for any adversary $\mathcal{A}$ making at most $t$ group operations[8]:*

$$\Pr\left[\mathcal{H} \xleftarrow{\$} \mathcal{D}_{\mathcal{H}}\,;\, s \xleftarrow{\$} \mathbb{G}\,;\, \tilde{\pi} \xleftarrow{\$} \mathcal{D}\,;\, P \leftarrow \mathcal{H}(\pi)\,;\, H \leftarrow s^P\,;\, \mathcal{A}^{\mathcal{H}}(s,H) = P\right] \;\leq\; \frac{4t-4}{2^\beta} + \frac{2^{2n}}{p} + \frac{3}{p}.$$

We remark that the condition $p - 2^n > 2^{2n}\sqrt{p}$ is verified for any $n$ such that $2^{4n+1} < p$, and that $\frac{2^{2n}}{p} \leq \frac{1}{\sqrt{p}} \leq 2^{-\mathfrak{K}}$ (for any cyclic group in which the discrete logarithm problem is $\mathfrak{K}$-bit hard, because of generic attacks).

*Remark 5.* This theorem may seem slightly useless since the adversary $\mathcal{A}$ anyway obviously has to make about $2^{-\beta}$ queries to $\mathcal{H}$ to solve the problem. But often such queries are faster than group operations and as already mentionned in Section 3.2, this does not work with multiple **Hash** queries, while counting the number of group operations still works.

The proof is inspired by the proof of Schnorr in [Sch01] but is given entirely, for the sake of completeness and rigor. We remark that the bound is not as tight as the one obtained by Schnorr, but this is sufficient for our purpose.

*Proof.* First of all, let us use an additive notation (as for graded rings): for any $x \in \mathbb{Z}_p$ and $u, v \in \mathbb{G}$: $x \odot u = x^u$, $u \oplus v = u \cdot v$. Let us also suppose that $t < 2^{n-1}$, since the bound is trivial to prove otherwise (because $\beta \leq n$ and so $(4t-4)/2^\beta > 1$).

In the generic group model, each element is represented by a random string and operations $\oplus$ and $\odot$ are performed by two oracles[9]. $t$ is the number of oracle queries.

Let us write $\mathsf{Exp}_0$ the experiment implicitely considered in the theorem and depicted in Figure 3. The probability considered in the theorem (we want to bound) is just $\mathsf{Succ}_{\mathsf{Exp}_0}$. The adversary queries to the oracles $\oplus$ and $\odot$ can be seen as polynomials:

$$Q_i := x_i \odot s \ominus y_i \odot H,$$

with unknowns $s$ and $H$, and coefficients $x_i$ and $y_i$ ($1 \leq i \leq t$).

Let $Q'_i$ be the polynomial $Q_i$ where $H$ is replaced by $P \odot s = s^P$. Let us then consider the experiment $\mathsf{Exp}_1$ which is similar to $\mathsf{Exp}_0$ except that the oracles return an independent random string for each distinct $Q_i$, as if $H$ and $s$ were independent (the resulting oracles are denoted $\oplus'$ and $\odot'$ in Figure 3), and, except that, at the end, we make the adversary win the experiment if $Q_i \neq Q_j$ but $Q'_i = Q'_j$. In other words, we make the adversary win, if $Q_i$ and $Q_j$ are not formally equal but become equal when $H$ is replaced by $P \odot s$. This is called the event $E$. $\oplus'$ and $\odot'$ return the same string for two (formally) equal polynomials and in the sequel, we suppose, without loss of generality, that all polynomials $Q_i$ are formally distinct.

[8] More precisely, making at most $t$ oracle queries in the generic group models, which is stronger in our case. Precise definition of oracle queries are given in the proof.

[9] One oracle for $\oplus$ would be sufficient since $x \odot u$ can be computed using $\oplus$. But allowing direct calls to the $\odot$ oracle strengthens our theorem. Actually, we can even allow direct calls to an oracle able to compute expressions of the form $x \odot s \ominus y \odot H$ ($x, y$ being inputs of this oracle).

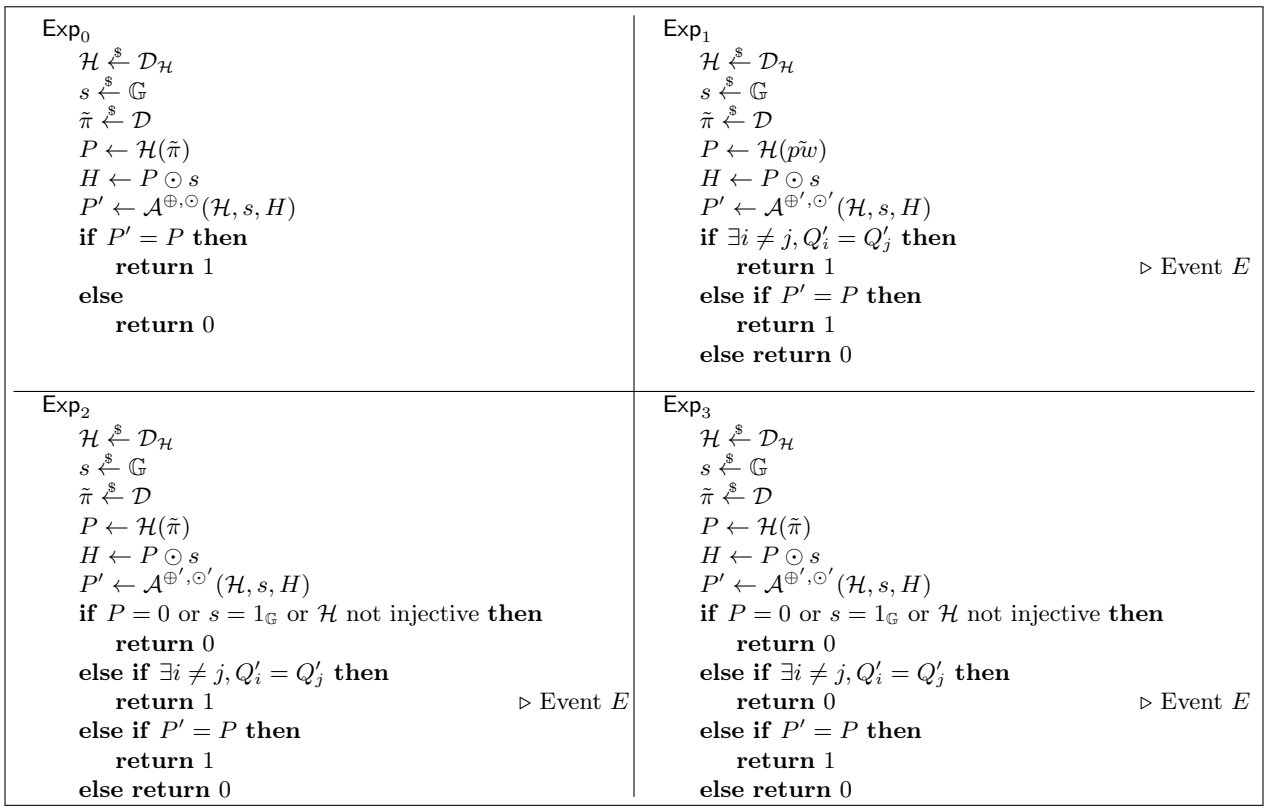

Fig. 3: Experiments for Theorem 4

If the event $E$ does not happens $\mathsf{Exp}_1$ and $\mathsf{Exp}_0$ are identical, and otherwise, the adversary always wins in $\mathsf{Exp}_1$. Clearly:

$$\mathsf{Succ}_{\mathsf{Exp}_0} \leq \mathsf{Succ}_{\mathsf{Exp}_1}. \tag{1}$$

Let us now consider the experiment $\mathsf{Exp}_2$ which is similar to $\mathsf{Exp}_1$, except we abort (the adversary loses) when $P = 0$ or $s = 1_{\mathbb{G}}$ or $\mathcal{H}$ is not injective. We have:

$$\mathsf{Succ}_{\mathsf{Exp}_1} - \mathsf{Succ}_{\mathsf{Exp}_2} \leq \frac{2}{p} + \frac{2^{2n}}{p}, \tag{2}$$

since $\Pr[\, P = 0 \text{ or } s = 1_{\mathbb{G}} \,] \leq 2/p$ and the probability that $\mathcal{H}$ is not injective is:

$$\Pr[\, \exists \pi \neq \pi',\, \mathcal{H}(\pi) = \mathcal{H}(\pi') \,] \leq \frac{2^n(2^n+1)}{2}\frac{1}{p} \leq \frac{2^{2n}}{p}.$$

Let us now consider the experiment $\mathsf{Exp}_3$ which is similar to $\mathsf{Exp}_2$, except that when the event $E$ happens, the adversary loses the experiment. Clearly, we have:

$$\mathsf{Succ}_{\mathsf{Exp}_3} \leq 2^{-\beta}, \tag{3}$$

the probability of success of $\mathcal{A}$ in $\mathsf{Exp}_3$ is at most $2^{-\beta}$, since $\mathcal{H}$ is injective and the password is not used in this game (except to define the event $E$, but this is done at the end of the experiment and is not known to the adversary), and so the adversary may only guess the password.

Let us now bound $\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3}$. The two experiments are identical except when two polynomials $Q_i$ and $Q_j$ $(i \neq j)$ are equal when $H$ is replaced by $s^P$. This happens if and only if $(x_i - x_j) = (y_i - y_j)P$. If $(x_i - x_j) = (y_i - y_j)P$, $y_i \neq y_j$ otherwise $x_i = x_j$ and $Q_i = Q_j$ (as polynomials), and this case is already excluded. Let $S = \{\frac{x_i - x_j}{y_i - y_j} \mid i \neq j\}$ and $T = S \cap \mathcal{H}(\{0,1\}^n)$, where $\frac{x_i - x_j}{y_i - y_j}$ is arbitrarily set to 0 if $y_i = y_j$. Then, from the adversary point of view, $\mathsf{Exp}_0$ and $\mathsf{Exp}_1$ are distinct only if $P \in S$. Since $P \in \mathcal{H}(\{0,1\}^n)$ this is equivalent to $P \in T$. Since $\mathcal{H}$ is injective, this happens with probability at most $|T|/2^{\beta}$, with $|T|$ the size of the set $T$. Therefore, we have[10]:

$$\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3} \leq \frac{|T|}{2^{\beta}}. \tag{4}$$

[10] This equation is not rigorous and is informal since $T$ depends on the actual execution the experiment, and so is not well defined. But this gives the idea why it is important to try to bound $|T|$.

It remains to bound $|T|$. Let us first give a quick summary of the proof, before giving the details. First of all, there is at most $t(t+1)/2$ values $\frac{x_i-x_j}{y_i-y_j}$, so $|T| \leq |S| \leq t(t+1)/2$. But this bound is not sufficient here. To get a better bound, we will bound the number $M$ of sets $\mathcal{H}(\{0,1\}^n)$ such that there exists some set $S$ (defined by some $(x_i, y_i)$) such that $T = \mathcal{H}(\{0,1\}^n) \cap S$ has size at least $m = 4t-4$. More precisely, we will show that $M$ is negligible compared to the number of sets $\mathcal{H}(\{0,1\}^n)$ (which is $\binom{p}{2^n}$). This will prove that with high probability over the choice of $\mathcal{H}$, any set $T$ (defined by some set $S$, itself defined by a sequence of $(x_i, y_i)$) contains at most $4t-4$ elements, which concludes the proof.

Here are the details. We first remark that $S$ is defined by the sequence of $t$ distinct pairs $(x_i, y_i)$. Furthermore, if the $x_i$'s and $y_i$'s are replaced by $x'_i = \lambda x_i + x'$ and $y'_i = \lambda y_i + y'$ (with $\lambda \in \mathbb{Z}_p \setminus \{0\}$ and $x', y' \in \mathbb{Z}_p$), the $x'_i$'s and $y'_i$'s yield the same set $S$ as the $x_i$'s and $y_i$'s. So there are at most $p^2(p^2-1)\cdots(p^2-t)/((p-1)p^2) \leq p^{2t-3}$ sets $S$.

Since $\mathcal{H}$ is injective, the set $\mathcal{H}(\{0,1\}^n)$ is a random set of $2^n$ (distinct) elements in $\mathbb{Z}_p$. Therefore there are $\binom{p}{2^n}$ such sets. Let us now count the number $M$ of such sets $\mathcal{H}(\{0,1\}^n)$ such that there exists a set $S$ (defined by some $x_i$'s and $y_i$'s) such that $T = \mathcal{H}(\{0,1\}^n) \cap S$ has size at least $m$ ($m$ will be chosen later in the proof). This number $M$ is at most the number of sets $S$ times the number of subsets of size $m$ of $S$ times the number of subsets of $\mathbb{Z}_p$ of size $2^n - m$:

$$M \leq p^{2t-3} \binom{\frac{t(t+1)}{2}}{m} \binom{p}{2^n - m}.$$

And so, using the inequality $k! \geq e^k/k^k$ for any positive integer $k$, the probability that $S'$ has at least $m$ elements is at most:

$$\begin{aligned}
\frac{M}{\binom{p}{2^n}} &\leq p^{2t-3} \cdot \binom{\frac{t(t+1)}{2}}{m} \cdot \frac{\binom{p}{2^n-m}}{\binom{p}{2^n}} \\
&\leq p^{2t-3} \cdot \frac{(t(t+1)/2)^m}{m!} \cdot \frac{(2^n)!}{(2^n-m)!} \cdot \frac{1}{(p-(2^n-m+1))\cdots(p-2^n)} \\
&\leq p^{2t-3} \cdot \frac{(t(t+1))^m \cdot e^m}{2^m \cdot m^m} \cdot 2^{nm} \cdot \frac{1}{(p-2^n)^m} \\
&\leq p^{2t-3} \cdot \left(\frac{t(t+1)\cdot e \cdot 2^n}{2 \cdot m \cdot (p-2^n)}\right)^m
\end{aligned}$$

If we take $m = 4t-4$, since $t \leq 4t-4$ (for $t \geq 2$, but for $t = 1$ no collision is possible) and $t+1 \leq 2^{n-1}$, we get:

$$\begin{aligned}
\frac{M}{\binom{p}{2^n}} &\leq p^{2t-3} \cdot \left(\frac{t(t+1)\cdot e \cdot 2^n}{2(4t-4)\cdot p - 2^n}\right)^{4t-4} \\
&\leq p^{2t-3} \cdot \left(\frac{t}{4t-4} \cdot \frac{e}{4} \cdot \frac{2(t+1)\cdot 2^n}{p-2^n}\right)^{4t-4} \\
&\leq p^{2t-3} \cdot \left(1 \cdot 1 \cdot \frac{2^{2n}}{p-2^n}\right)^{4t-4} \\
&\leq p^{2t-3} \cdot \left(\frac{1}{\sqrt{p}}\right)^{4t-4} \\
&\leq \frac{1}{p},
\end{aligned}$$

where the last-but-one inequality comes from the fact $p - 2^n > 2^{2n}\sqrt{p}$.

From Equation (4) (or more precisely, from an analysis of $\mathsf{Exp}_2$ and $\mathsf{Exp}_3$), we get:

$$\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3} \leq \frac{4t-4}{2^\beta} + \frac{1}{p}, \tag{5}$$

since either $|T|$ is bounded by $m = 4t-4$, in which case the adversary wins with probability at most $\frac{m}{2^\beta}$, or $|T|$ is not, but this happens only with probability at most $\frac{1}{p}$. From Equations (1), (2), (5) and (3), we get:

$$\mathsf{Succ}_{\mathsf{Exp}_0} \leq \frac{4t-4}{2^\beta} + \frac{2^{2n-1}}{p} + \frac{3}{p}.$$

This concludes the proof. □

**Case with Multiple Hash Queries.** Let $q$ be the number of **Hash** queries and let $s_1, \ldots, s_q$ and $\tilde{\pi}_1, \ldots, \tilde{\pi}_q$ be respectively the $q$ salts and $q$ passwords chosen by **Hash**. Let $P_k = \mathcal{H}(\tilde{\pi}_k)$ and $H_k = s_k^{P_k}$.

The adversary queries to the oracles $\oplus$ and $\odot$ can be seen as polynomials:

$$Q_j \coloneqq \bigoplus_{k=1}^{q} x_{k,j} \odot s_k \ominus \bigoplus_{k=1}^{q} y_{k,j} \odot H_k,$$

with unknowns $s_k$ and $H_k$, and coefficient $x_{k,j}$ and $y_{k,j}$ ($1 \leq j \leq t$ and $1 \leq k \leq q$).

Let us now consider the same experiments as in the case with only one **Hash** query: $Q'_j$ is now the polynomial $Q_j$ where $H_k$ has been replaced by $P_k \odot s_k$ (for all $k$); and in $\mathsf{Exp}_2$ and $\mathsf{Exp}_3$, we abort when one of the $P_k$ is 0 or one of the $s_k$ is $1_{\mathbb{G}}$ (which happens with probability at most $2q/p$ instead of $2/p$).

Everything works as before, except the bound on $\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3}$ (which was the core of the previous proof). The two experiments are identical except when $Q'_j = Q'_{j'}$ for some $j \neq j'$ (event $E$). This happens if and only if $x_{k,j} - x_{k,j'} = (y_{k,j} - y_{k,j'})P_k$ for all $k$. Let us now consider the graph whose nodes are the polynomials $Q_j$. There is an edge, labeled by a set $S_{j,j'} = \{(k_{j,j',1}, P'_{j,j',1}), \ldots, (k_{j,j',q'}, P'_{j,j',q'})\}$, with $1 \leq k_{j,j',1} < \cdots < k_{j,j',q'} \leq q$ and $P'_j \in \mathbb{Z}_p$, between $Q_j$ and $Q'_j$, with $j \neq j'$, if:

$$Q_j \ominus Q'_{j'} = \bigoplus_{(k,P') \in S_{j,j'}} ((x_{k,j} - x_{k,j'}) - (y_{k,j} - y_{k,j'})P') \odot s_k;$$

in other words, if $Q'_j = Q'_{j'}$ when $P'_{j,j',i} = P_{k_{j,j',i}}$ for all $i = 1, \ldots, q'$, in which case we say that $S_{j,j'}$ is *compatible* with $(P_k)$. We remark that $\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3}$ is the probability that some $S_{j,j'}$ is compatible with $(P_k)$.

Let us first handle the case when some non-singleton label $S_{j,j'}$ is compatible with $(P_k)$. There are at most $t(t+1)/2$ distinct $S_{j,j'}$, and the probability that $(P_k)$ is compatible with a set $S_{j,j'}$ containing at least two elements is at most $2^{-2\beta}$. So this first case arrives with probability at most $\frac{t(t+1)}{2^{2\beta+1}}$.

Let us now only consider singleton labels $S_{j,j'}$ and remove all edges with non-singleton labels in the previous graph. This graph represents the possible collisions (on the $Q'_j$'s) involving only one **Hash** query. More precisely, if we set:

$$S_i \coloneqq \{P' \mid \text{there exists an edge labeled } \{(i, P')\} \text{ in the graph}\}$$

the event $E$ happens either for some non-singleton label $S_{j,j'}$ (case already handled) or when $P_i \in S_i$ for some $i$. The graph may not be connected. In each connected component $C$ of this graph, we chose an arbitrary node $Q_{j^*_C}$ and replace every polynomial $Q_j$ in $C$ by $Q_j - Q_{j^*_C}$ (and update accordingly the $x_{k,i}$'s and $y_{k,i}$'s). We also remove all connected components with only one node. Let $\mathcal{Q}$ be the set of remaining polynomials. All these modifications do not change the sets $S_i$, and, in addition, we have:

$$S_i \subseteq S'_i \coloneqq \left\{ \frac{x_{i,j} - x_{i,j'}}{y_{i,j} - y_{i,j'}} \;\middle|\; (Q_j, Q_{j'}) \in \mathcal{Q}^2 \text{ and } y_{i,j} \neq y_{i,j'} \right\},$$

and each set $S'_i$ actually corresponds to the set $S$ in the proof for one **Hash** query, for the polynomials $R_{i,j} = x_{i,j} \odot s_i \ominus y_{i,j} \odot H_i$ (for all $j$ such that $P_j \in \mathcal{Q}$). Let $\mathcal{Q}_i = \{R_{i,j} \mid Q_j \in \mathcal{Q}\}$ and $t_i = |\mathcal{Q}_i|$ be the number of distinct polynomials $R_{i,j}$. For the sequel, we suppose $t_i \geq 1$ for all $i$, without loss of generality (since we can always ignore the ones with $t_i = 0$, this only makes the number of queries $q$ smaller).

It remains now to prove two points:

1. with probability at least $\frac{t}{p}$ over the choice of $\mathcal{H}$, no set $\mathcal{Q}_i$ yields a set $T'_i \coloneqq S'_i \cap \mathcal{H}(\{0,1\}^n)$ of size greater than $4t_i - 4$;
2. $(t_1 - 1) + \cdots + (t_q - 1) \leq t$.

Indeed, if $P_i \in S_i$, then $P_i \in T'_i$, and $P_i$ are distributed according to a distribution of min-entropy $\beta$ (because $\mathcal{H}$ is supposed to be injective here). Therefore, the probability that $P_i \in T'_i$ is at most $(4t_i - 4)/2^\beta$ when $|T'_i| \leq 4t_i - 4$, and:

$$\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3} \leq \frac{t(t+1)}{2^{2\beta+1}} + \sum_{i=1}^{q} \frac{4t_i - 4}{2^\beta} + \frac{t}{p} \leq \frac{t^2}{2^{2\beta}} + \frac{4t}{2^\beta} + \frac{t}{p}.$$

Since a success probabiblity is at most 1, and since $t \geq 2^\beta$ implies $(4t - 4)/2^\beta \geq 1$, we get:

$$\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3} \leq \frac{5t}{2^\beta} + \frac{t}{p}.$$

Finally, we have:

$$\mathsf{Succ}_{\mathsf{Exp}_0} \leq \frac{5t}{2^\beta} + \frac{2^{2n}}{p} + \frac{t + 2q}{p}.$$

It remains to prove our two points. The first point comes directly from the union bound over all the possible sizes of $\mathcal{Q}_i$ and the previous proof: from the previous proof, the probability of any set of at most $t'$ polynomials yield a corresponding set $T'$ of size greater than $4t' - 4$ is at most $\frac{1}{p}$, therefore the probability that there exists some $1 \leq t' \leq t$ and some set of at most $t'$ polynomials verifying the above property is at most: $\frac{t}{p}$.

To prove the second point, let us show that in any connected component $C$ with $c$ vertices, there are at most $c - 1$ distinct tuples $(i, x_{i,j}, y_{i,j}) \neq (i, 0, 0)$ (for $Q_j \in C$). Let the depth of a node $Q_j$ be the distance between the previously arbitrarily selected node $Q_{j^*_C}$ (now equal to 0 since all $Q_j$'s have been replaced by $Q_j - Q_{j^*_C}$) and the node $Q_j$. We prove by recursion on $d \geq 0$ that there are at most $c_d - 1$ distinct tuples $(i, x_{i,j}, y_{i,j}) \neq (i, 0, 0)$ for $Q_j \in C$ of depth at most $d$, where $c_d$ is the number of nodes of depth at most $d$. The only node of depth 0 is $Q_{j^*_C}$ and for all $i$, $(i, x_{i,j^*_C}, y_{i,j^*_C}) = (i, 0, 0)$, so the property is true for $d = 0$. Let us suppose the property true for $d$ and prove it for $d + 1$: each node $Q_j$ of depth $d + 1$ is such that there exists an edge $S_{j,j'} = \{(i, P')\}$ between $Q_j$ and some node $Q_{j'}$ of depth $d$; and so for all $i' \neq i$, $(i', x_{i',j}, y_{i',j}) = (i', x_{i',j'}, y_{i',j'})$. Therefore each node $Q_j$ adds at most one new distinct $(i, x_{i,j}, y_{i,j}) \neq (i, 0, 0)$, and there are at most $c_d - 1 + (c_{d+1} - c_d) = c_{d+1} - 1$ distinct $(x_{i,j}, y_{i,j}) \neq (0, 0)$ for $Q_j \in C$ of depth at most $d$ (since there are $c_{d+1} - c_d$ nodes of depth $d + 1$). This concludes the proof that in any connected component $C$ with $c$ vertices, there are at most $c - 1$ distinct tuples $(i, x_{i,j}, y_{i,j}) \neq (i, 0, 0)$ (for $Q_j \in C$).

Since there are $t$ nodes, there are at most $t - 1$ distinct tuples $(i, x_{i,j}, y_{i,j}) \neq (i, 0, 0)$ for $Q_j \in \mathcal{Q}$ a node of the graph, and since there are $q$ distinct tuples $(i, x_{i,j}, y_{i,j}) = (i, 0, 0)$, in total, there are at most $t - 1 + q \leq t + q$ distinct tuples $(i, x_{i,j}, y_{i,j})$. Finally, as $t_1 + \cdots + t_q$ is at most this number of distinct pairs:

$$t_1 + \cdots + t_q \leq t + q,$$

and:

$$(t_1 - 1) + \cdots + (t_q - 1) \leq t.$$

### C.2 Security of our Password Hashing Scheme Based on Multi-Linear Maps

In this section, we handle the case $q \geq 2$ in the proof of the tight one-wayness of our password hashing scheme based on multi-linear maps.

Let us consider the graph whose nodes are the polynomials $Q_j$. There is an edge, labeled by a set $S_{j,j'} = \{(k_{j,j',1}, \pi_{j,j',1}), \ldots, (k_{j,j',q'}, \pi_{j,j',q'})\}$, with $1 \leq k_{j,j',1} < \cdots < k_{j,j',q'} \leq q$ and $\pi_{j,j',i} \in \{0,1\}^n$, between $Q_j$ and $Q_{j'}$ with $j \neq j'$ if

$$Q_j \ominus Q_{j'} = \bigoplus_{(k,\pi) \in S_{j,j'}} c_{j,j',k} \odot (a_{1,\pi[1]} \odot \cdots \odot a_{n,\pi[n]} \odot s_k \ominus H_k),$$

with $c_{j,j',k}$ a non-zero constant (for all $k = 1, \ldots, q'$); or in other words, if $Q_j$ and $Q_{j'}$ would be equal if $H_k$ is computed correctly ($H_k = a_{1,\tilde{\pi}_k[1]} \odot \cdots \odot a_{n,\tilde{\pi}_k[n]} \odot s_k$) and $\pi_{j,j',i} = \tilde{\pi}_{k_{j,j',i}}$ for all $i = 1, \ldots, q'$, in which case we say that $S_{j,j'}$ is *compatible* with $(\tilde{\pi}_k)_k$. We remark that, with high probability, the adversary can distinguish the simulated oracles from the real oracles, if and only if one label $S_{j,j'}$ of the graph is compatible with $(\tilde{\pi}_k)_k$.

Let us now bound the number of distinct singleton labels. For that, we can use exactly the same proof as the one used to bound the number of labels in the case $q = 1$. We can indeed consider the graph where all edges with non-singleton labels are removed and where we arbitrarily remove edges in such a way there is only one edge per distinct label in the initial graph.

Let us now proceed by contradiction: we assume there is a minimal cycle $Q_{j_1}, \ldots, Q_{j_\ell}, Q_{j_{\ell+1}} = Q_{j_1}$ with edges of labels $\{(k_1, \pi_1)\}, \ldots, \{(k_\ell, \pi_\ell)\}$ in this new graph. We get:

$$0 = \bigoplus_{u=1}^{\ell} (Q_{j_u} \ominus Q_{j_{u+1}}) = \bigoplus_{u=1}^{\ell} c_{j_u,j_{u+1}} \odot (a_{1,\pi_u[1]} \odot \cdots \odot a_{n,\pi_u[n]} \odot s_{k_u} \ominus H_{k_u})$$

which is impossible since all labels $(k_1, \pi_1), \ldots, (k_\ell, \pi_\ell)$ are distinct and so all the monomials $a_{1,\pi_u[1]} \odot \cdots \odot a_{n,\pi_u[n]} \odot s_{k_u}$ (for $u = 1, \ldots, \ell$) are distinct. Therefore, the new graph has no cycle and the number of distinct singleton labels is at most $t$.

Let $m_k$ is the numbers of distinct singleton labels of the form $\{(k, \cdot)\}$ and $m'$ the number of distinct non-singleton labels. From the previous analysis, $m_1 + \cdots + m_\ell \leq t$. In addition, $m'$ cannot be more than the maximum number of edges in a graph with $t$ nodes, and is thus less than $t^2$. In addition, for any non-singleton label $S_{j,j'}$, the probability that $S_{j,j'}$ is compatible with $(\tilde{\pi}_k)_k$ is $2^{-2\beta}$. Therefore, the probability for the adversary to win is at most:

$$\frac{m_1}{2^\beta} + \cdots + \frac{m_\ell}{2^\beta} + \frac{m'}{2^{2\beta}} \leq \frac{t}{2^\beta} + \frac{t^2}{2^{2\beta}}.$$

Since a probability is never greater than 1 and since $t \geq 2^\beta$ implies $t/2^\beta \geq 1$, the probability for the adversary to win is at most $2t/2^\beta$.

### C.3 Relation between the BPR-like Model and the Related-Password Model

Let us show why a VPAKE protocol secure in the related-password model is also secure in the BPR-like model. The four properties of a password hashing scheme are written on italics to emphasize on where there are needed.

In $\mathbf{G}_{\mathtt{ideal}}$, if one cannot extract a valid pair $(s, H)$ or a valid pre-hash value $P$, then the two cases $b = 0$ and $b = 1$ are perfectly indistinguishable for the adversaries, since `Test` queries always return random values or $\perp$.

Let us now bound the probability that we can extract a valid pair $(s, H)$ or a valid pre-hash $P$. This will directly bound the advantage $\mathsf{Adv}_{\mathtt{ideal}}(\mathfrak{K}, \mathcal{A})$ of a polynomial-time adversary $\mathcal{A}$ in the game $\mathbf{G}_{\mathtt{ideal}}$. If the adversary did not ask any `Test` query on a user with a corrupted password hash, he sees absolutely no information on passwords of users involved in tested session. Since passwords are drawn from some distribution $\mathcal{D}$ of min-entropy $\beta$, it follows from the *entropy preservation* and the *pre-hash entropy preservation* that the probability to extract a valid pair $(s, H)$ or a valid pre-hash $P$ (respectively) is at most $q_s \times 2^{-\beta} + \mathrm{negl}(\mathfrak{K})$, with $q_s$ the number of times Extract is called, i.e., the number of `Test` queries on active sessions (thanks to the union bound).

Otherwise, in addition to the probability for `Test` queries on users without corrupted password hash, we have to consider the probability to extract a valid pre-hash value $P$ for a corrupted password hash $H$. This probability is bounded by $q_S \times \mathsf{Adv}_{\mathtt{snd}}(\mathcal{B}', \mathfrak{K}) + \mathsf{Adv}_{\mathtt{one\text{-}way}}(\mathcal{B}, \mathfrak{K}) = \mathsf{Adv}_{\mathtt{one\text{-}way}}(\mathcal{B}, \mathfrak{K}) + \mathrm{negl}(\mathfrak{K})$, with $\mathcal{B}$ and $\mathcal{B}'$ adversaries running approximatively in the same time as $\mathcal{A}$ (plus the time of simulating the oracles of the games and of running the Extract algorithm). Indeed, the *second pre-image resistance* ensures that the adversary $\mathcal{A}$ cannot find a pre-hash value $P'$ from some corrupted password hash $H_S$ of some server $S$ different from the pre-hash value $P_S = \mathsf{PPreHash}(\mathsf{param}, \pi_S)$

corresponding to the password $\pi_S$. And the *tight one-wayness* ensures that finding any pre-hash value $P_S = \mathsf{PPreHash}(\mathsf{param}, \pi_S)$ for any corrupted server $S$ takes at least about the same time as brute-forcing the password.

This concludes the proof since the advantage $\mathsf{Adv}_{\mathtt{real}}(\mathfrak{K}, \mathcal{A})$ of any adversary $\mathcal{A}$ in the real game is essentially upper-bounded by $\mathsf{Adv}_{\mathtt{ideal}}(\mathfrak{K}, \mathcal{A}) + \mathrm{negl}(\mathfrak{K})$.

### C.4 Security of the Generic Two-Round VPAKE

This proof is close to the proof from [GL03]. However, due to the fact our model is slightly stronger and takes care of the case where two incompatible users are partnered, we need to be a little more careful.

The proof consists in proving that the real attack game $\mathbf{G}_{\mathtt{real}} = \mathbf{G}_0$ is indistinguishable from the ideal one $\mathbf{G}_{\mathtt{ideal}}$ described in Section 4.2.

The $\mathsf{Extract}$ algorithm just decrypts the ciphertext sent by the corresponding user under the corresponding label using the secret key associated to the public key $\mathsf{pk}$ in the CRS, and outputs the associated plaintext (together with the salt sent by the user if the user was a server).

Let us consider a polynomial time adversary $\mathcal{A}$. The proof is done by a series of games $\mathbf{G}_i$, and $\mathsf{Adv}_i(\mathcal{A}, \mathfrak{K})$ is the advantage of $\mathcal{A}$ in the game $\mathbf{G}_i$. For the sake of simplicity $\mathfrak{K}$ is often implicit in this proof. In particular, $\mathrm{negl} = \mathrm{negl}(\mathfrak{K})$.

We separate $\mathtt{Send}$ queries in three types:

- $\mathtt{Send}_0(S^i, C^j, \mathtt{Start})$ queries which enable an adversary to ask $C^j$ to initiate the protocol with $S^i$ and which return the first flow for $C^j$ and $S^i$.
- $\mathtt{Send}_1(C^i, S^j, m)$ queries which enable an adversary to send the first flow for $C^i$ and $S^j$ and which return the second flow answered back by $S^j$;
- $\mathtt{Send}_2(S^i, C^j, m)$ queries which enable an adversary to send the second flow for $C^j$ and $S^i$ and which return nothing but set the session key $K$ of $S^i$.

**Game $\mathbf{G}_1$:** We first modify the way $\mathtt{Execute}$ queries between two compatible users are answered. Since the hashing keys are known, we compute the common session key as

$$K = K_S = K_C = \mathsf{Hash}(\mathsf{hk}_S, L'_{(s_S,H_S)}, (\ell_C, c_C)) \times \mathsf{Hash}(\mathsf{hk}_C, L_{H_S}, (\ell_S, c_S)),$$

which does not change anything thanks to the correctness of the $\mathsf{SPHF}$s. In addition, we replace $c_S$ and $c_C$ by encryptions of the dummy password 0. This is indistinguishable from $\mathbf{G}_0$ under the $\mathtt{IND\text{-}CPA}$ property of the encryption scheme, for each $\mathtt{Execute}$ query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A})| \leq \mathrm{negl}$.

**Game $\mathbf{G}_2$:** We modify again the way $\mathtt{Execute}$ queries between two compatible users are answered: we replace the common session key by a truly random value. Since the languages are not satisfied, the smoothness guarantees indistinguishability: $|\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})| \leq \mathrm{negl}$.

**Game $\mathbf{G}_3$:** We now modify the way $\mathtt{Execute}$ between two incompatible users are answered: we replace both session keys

$$K_C = \mathsf{ProjHash}(\mathsf{hp}_S, L'_{(s_S,H_S)}, (\ell_C, c_C), r_C) \times \mathsf{Hash}(\mathsf{hk}_C, L_{H_S}, (\ell_S, c_S))$$
$$K_S = \mathsf{Hash}(\mathsf{hk}_S, L'_{(s_S,H_S)}, (\ell_C, c_C)) \times \mathsf{ProjHash}(\mathsf{hp}_C, L_{H_S}, (\ell_S, c_S), r_C)$$

(for the client and the server) by two independent truly random values. Thanks to the smoothness of the $\mathsf{SPHF}$s, $\mathsf{Hash}(\mathsf{hk}_C, L_{H_S}, (\ell_S, c_S))$ and $\mathsf{Hash}(\mathsf{hk}_S, L'_{(s_S,H_S)}, (\ell_C, c_C))$ are (close to) completely independent random values. And we have $|\mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A})| \leq \mathrm{negl}$.

**Game $\mathbf{G}_4$:** We modify again the way $\mathtt{Execute}$ queries between two incompatible users are answered: we replace $c_S$ and $c_C$ by encryptions of the dummy password 0. This is indistinguishable under the $\mathtt{IND\text{-}CPA}$ property of the encryption scheme, for each $\mathtt{Execute}$ query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A})| \leq \mathrm{negl}$.

**Game $\mathbf{G}_5$:** We now modify the way the $\mathtt{Send}_2$ queries are answered, by using a decryption oracle, or alternatively knowing the decryption key. More precisely, when a message $(s, \mathsf{hp}_S, c_S)$ is sent, in the name of some server instance $S^i$ and to some client instance $C^j$, four cases can appear:
- it is not a message generated by $S$ (via a $\mathtt{Send}_1$ query) after receiving the first flow $(\mathsf{hp}_C, c_C)$ sent by $C^j$ (via a $\mathtt{Send}_0$ query), then we first decrypt[11] the ciphertext to get the hash value $H$ used be the adversary:
  1. if $H = \mathsf{PFullHash}(\mathsf{param}, s, \pi_C)$ (i.e., if the password $\pi_C$ is compatible with $(s, H)$) then we make the adversary wins and continue the game as in $\mathbf{G}_{\mathtt{ideal}}$;
  2. otherwise, we choose the session key $K$ at random;
- if it is such a message from some $S^{i'}$, then $C^j$ is partnered with $S^{i'}$:
  3. if $S$ and $C$ are compatible, we set the session key of $C^j$ equal to the one already computed by $S^{i'}$;
  4. otherwise, we choose a random session key $K$.

The change in the first case can only increase the advantage of the adversary, while the changes in the second and fourth cases are indistinguishable under the smoothness of the KVSPHF and thus only increase the advantage of the adversary by a negligible term. The change in the third case does not change the advantage of the adversary. Therefore, we have: $\mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A})+\mathrm{negl}$.

**Game $\mathbf{G}_6$:** In $\mathbf{G}_5$, we remark that we do not need to know the random coins used by the ciphertext $c_C$ generated in response to a $\mathtt{Send}_0$ query. So, we can simply encrypt the dummy password 0 instead of the correct password $\pi_C$ in all ciphertexts $c_C$, generated as responses to $\mathtt{Send}_0$ queries. This is indistinguishable under the IND-CPA property of the encryption scheme: $|\mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A})| \leq \mathrm{negl}$.

**Game $\mathbf{G}_7$:** We modify now the way we answer the $\mathtt{Send}_1$ queries, similarly to what we have done for the $\mathtt{Send}_2$ queries in $\mathbf{G}_5$. More precisely, when a message $(\mathsf{hp}_C, c_C)$ is sent, in the name of some client instance $C^i$ and to some server instance $S^j$, three cases can appear:
- it has been generated (altered) by the adversary, then we first decrypt the ciphertext to get the pre-hash value $P$ used be the adversary:
  - if $H_S = \mathsf{PHash}(\mathsf{param}, s_S, P)$ (with $H_S$ and $s_S$ the hash value and salt of the server $S$) then we make the adversary wins and continue the game as in $\mathbf{G}_{\mathtt{ideal}}$;
  - otherwise, we choose the session key $K$ at random;
- it is a replay of a previous flow sent by the simulator, then, in particular, we know the hashing key $\mathsf{hk}_C$ associated with $\mathsf{hp}_C$, and we compute the session key $K$ using the hashing key $\mathsf{hk}_C$ instead of $\mathsf{hp}_C$ together with the random coins used in the ciphertext $c_S$ ($c_C$ being sent as answer to the $\mathtt{Send}_1$ query).

The change in the first case can only increase the advantage of the adversary, while the change in the second case is indistinguishable under the smoothness of the GLSPHF and thus only increases the advantage of the adversary by a negligible term. The change in the third case does not change the advantage of the adversary. Therefore, we have: $\mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A}) + \mathrm{negl}$.

**Game $\mathbf{G}_8$:** We modify the way we answer the $\mathtt{Send}_1$ queries for replayed messages. More precisely, when a message $(\mathsf{hp}_C, c_C)$ is replayed by the adversary, we choose the session key $K$ at random. This is indistinguishable, since in this case, $c_C$ is an encryption of the dummy password 0 and, thanks to the smoothness of the KVSPHF, the hash value of $c_C$ under $\mathsf{hk}_S$ looks completely random (given only $\mathsf{hp}_S$): $|\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})| \leq \mathrm{negl}$.

We remark that, in this game, all session keys returned by the $\mathtt{Test}$ queries are completely independent and random (except for partnered compatible users, for which they are equal).

**Game $\mathbf{G}_9$:** We remark that we do not need to know the random coins used by the ciphertexts $c_S$ generated in response to a $\mathtt{Send}_1$ query, in the previous game. So, we can simply encrypt the dummy password 0 instead of the correct password $\pi_S$ in all ciphertexts $c_S$, generated as responses

[11] Later we will use the IND-CCA game over all ciphertexts we generate in the name of a server. Notice that decrypting $c_S$ under label $\ell_S = ((C, S), c_C, \mathsf{hp}_C, s_S, \mathsf{hp}_S)$ is authorized in this IND-CCA game thanks to the label, which may only have been used by $S$ after receiving $(\mathsf{hp}_C, c_C)$.

to $\mathtt{Send}_1$ queries. This is indistinguishable under the IND-CCA property of the encryption scheme: $|\mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A})| \leq \text{negl}$.

This last game is exactly the game $\mathbf{G}_{\mathtt{ideal}}$. And we have proven that

$$\mathsf{Adv}_{\mathtt{real}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathtt{ideal}}(\mathcal{A}) + \text{negl},$$

which proves the security in the related-password model.

Note that corruptions are handled in the same way in $\mathbf{G}_{\mathtt{real}}$ and $\mathbf{G}_{\mathtt{ideal}}$. That is why we did not discuss corruptions in this proof.