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

Fabrice Benhamouda and David Pointcheval

ENS, Paris, France *

October 14, 2014

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 formally define some properties for the transform (password hashing) applied to the password for the storage on the server-side, for an efficient VPAKE use. A tight one-wayness is required to prevent improved password searches. We then enhance the Bellare-Pointcheval-Rogaway game-based model for PAKE to VPAKE protocols, in such a way that it allows a VPAKE protocol to be secure in the standard model. In addition, we show how to further extend this model to handle non-uniform and related passwords, both in case of PAKE and VPAKE. Finally, we propose very efficient constructions of password hashing and VPAKE protocols, which are nearly as efficient as the best PAKE protocols to date.

Keywords. Multilinear 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[1] 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 October 2013, and more recently in May 2014, many, if not all, passwords of the 148 million of eBay active accounts), 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.

* CNRS – UMR 8548 and INRIA

[1] At least, that is the most reasonable (and widespread) option nowadays. Smartcards are not common (for website authentication) and need to be carried with you every time you need to authenticate, while biometrics authentication is even less widespread and has its own problems (e.g., biometrics cannot be changed in case of corruption of server).

Of course, even when the server is never compromised, 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, if we do not consider server compromise, we are left with the 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.

If we want in addition to mitigate server compromise, which seems nowadays very important, then the server should only store verifiers for passwords and not the passwords themselves. Such protocols are called verifier-based password-authenticated key exchange (VPAKE) protocols. For security, a first requirement is that extracting a password should take a computation time linear in the number of possible passwords, when passwords are uniformly distributed. This corresponds to the time of the trivial off-line dictionary attack.

1.1 Contributions

In this paper, we essentially deal with VPAKE protocols in the standard model.

Our first main contribution is a new model for VPAKE, based on the BPR security model, with the real-or-random flavor (Sections 2 and 3). Contrary to the model in [GMR06], this model does not directly imply the use of an idealized model. In this new model, we define the notion of *password hashing* (Section 2) which formalizes the way in which a verifier can be computed from a password. More precisely, to ensure strong security in case of server compromise, we introduce the notion of *tight one-wayness* which says that extracting a password for one verifier chosen among a set of verifiers should take nearly as long as hashing 2^β passwords, where β is the min-entropy of the password distribution.

This rules out naive hashing, where the verifier is just a deterministic hash of the password. Indeed, in such a case, the adversary could find a password of one verifier in time $2^\beta/n$ if he gets n verifiers. For a server with hundreds of thousands of accounts, this would enable the adversary to quickly find the password of an account, and so to enter into the system and make huge damage. On the other hand, our stronger notion of password hashing offers an optimal security with regards to this kind of attacks: the time an adversary will take to find out a valid password is 2^β. Assuming a strict password policy to ensure reasonable min-entropy β, this would give enough time to the company to change all passwords or disable all accounts.

As an additional contribution, we extend our VPAKE model to deal with related passwords, which is a more realistic use case for PAKE, since humans often use related passwords for various servers. We also show that all our constructions meet this stronger security notion. This model can be simplified to get the first PAKE model dealing with related passwords, not in the UC framework. And PAKE from the Gennaro and Lindell [GL06] (GL) framework, which were proven secure in the BPR model, are actually secure in our stronger model, with possibly related passwords. The GL framework is currently one of the most efficient way of constructing PAKE in the standard model.

Our second main contribution is to propose two constructions of password hashing (Sections 2.3 and 5.2). They are algebraic, in the sense that they can easily be used with Smooth Projective Hashing Functions (SPHFs). Therefore, they nicely fit in the GL framework, and in Section 4, we show how to build a VPAKE from these password hashing schemes. Our constructions based on the first password hashing come in two flavors: one-round and two-round. Both are very competitive even with the most efficient PAKE protocols so far: the one-round version only requires the server to send two more group elements than the best one-round PAKE so far [BBC$^+$13c] (based on the framework of Katz and Vaikuntanathan [KV11]), while our two-round version even outperforms the GL framework (using a folklore improvement) and needs only three more group elements than the PAKE of Groce and Katz [GK10] (the most efficient BPR PAKE so far, also in two rounds). Computation complexities are also similar, and security assumptions are the same (plain DDH), assuming the tight one-wayness of the underlying password hashing scheme.

We highlight that, while the tight one-wayness of our proposed password hashing schemes is proven in idealized models (random oracle model and/or generic group model), the resulting VPAKE protocols (Section 4) are proven secure in the standard model, assuming the tight one-wayness of the underlying password hashing scheme. In other words, we can consider the tight one-wayness of the underlying password hashing schemes as assumptions. Justifying these assumptions using idealized models is a classical and reasonable approach in practice. Furthermore, it appears to be an open problem to construct (not necessarily algebraic) password hashing, with proven tight one-wayness under classical assumptions such as factorization, DDH, or collision-resistance.

We would like to point out that the proofs of tight one-wayness of our two password hashing schemes are actually quite challenging, due to the use of a strong notion of security for password hashing. In addition, they may be of independent interest. For example, one of our constructions involves multilinear 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. As just said, it is indeed not enough to prove the one-wayness of the hashing, but we additionally require the inversion to be at least linear in the number of passwords (tight one-wayness).

1.2 Related Work

Password-Authenticated Key Exchange (PAKE). PAKE protocols have been introduced by Bellovin and Merritt [BM92]. 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], using Smooth Projective Hash Functions (SPHFs).

A major issue in the BPR model 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 $2^{-\beta}$, with β the bit-length of the passwords, or the min-entropy. Another drawback of the BPR security model is the assumption on the password distribution: passwords are assumed to be independently and sometimes uniformly distributed, whereas humans use biased distributions, and definitely related passwords. In this paper, we overcome both these drawbacks by introducing a model that deals with related passwords, and 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] already managed to solve these two drawbacks by providing a model 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+13c, ABB+13]. However, due to the strong requirements of the UC framework, these schemes are much less efficient than those secure in the BPR framework (such as [GL06, BBC+13c]) and even less efficient than our first VPAKE construction.

Verifier-Based Password Authenticated Key Exchange (VPAKE). Regarding VPAKE, while many protocols have been designed and informally analyzed, no game-based security model has been proposed so far, but just an ideal functionality in the UC framework [GMR06] with a generic conversion (the so-called Ω-method). However, to model an off-line dictionary attack, after a server corruption, 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 dictionary attack (in UC-PAKE and UC-VPAKE). Unfortunately, because of that `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.

Subsequent Paper of Kiefer and Manulis [KM14]. In a recent paper [KM14], Kiefer and Manulis build upon a previous version of our paper and show how to register the verifier to the server without revealing the

password but still enforcing a given password policy via zero-knowledge proofs to ensure that passwords are strong enough.

While this is a nice contribution, we should point out that their password hashing is not tightly one-way (or tightly pre-image resistant) as claimed. Indeed, they do not hash the password through a random oracle, but directly convert it to an integer via a "structure-preserving map", for their zero-knowledge proofs to work. This makes possible to invert their hashing algorithm using a variant of the Pollard's kangaroo method described in [MT09]. More precisely, let $\mathbb{G}$ be a cyclic group of prime order p, and let g and h be two generators. The hash value of a password π (seen as an integer in $\{0, \ldots, N\}$ using their structure-preserving map) is $(H_1, H_2) \in \mathbb{G}^2$, with $H_1 = g^{s_P}$ and $H_2 = g^{s_P \pi} h^{s_H}$, where s_P is an unknown scalar in $\mathbb{Z}_p$, and s_H is a public salt in $\mathbb{Z}_p$. Recovering π can then be done in $(2 + o(1))\sqrt{N}$ group operations by computing the discrete logarithm of H_2/h^{s_H} in base H_1, which is much less than $N/2$ group operations required by brute-force.

We remark that our VPAKE scheme using our password hashing based on multilinear maps in Section 5 would certainly work with the construction in [KM14] and so would provide the first (though inefficient) VPAKE scheme with a tight one-way password hashing and a way to enforce password policy without revealing passwords.

2 Password Hashing

2.1 Notations

In this paper, the quality of an adversary $\mathcal{A}$ against a security notion 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 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. And we expect the inversion to require a time that is *provably* lower-bounded by a linear function of the size of the dictionary, or 2^β with β the min-entropy of the distribution.

Without salt, extracting a password from a set of N verifiers, would only require to perform $2^\beta/N$ hashing, instead of 2^β. That would be very dangerous in case of corruption of an authenticating PAKE server in a big company for example, where there are 10^5 to 10^6 employees, and getting access to any account of any employee may have disastrous consequences.

We may then wonder why not using the login as the salt. This would be very practical in our VPAKE scheme, because the salt could be used by the client when generating his first flow to the server, while if the salt is unpredictable, this would be impossible. However, doing so is also very dangerous. First, it enables an attacker to easily check if a user uses the same password in two (corrupted) databases or in the same database at two different time periods. Second, it allows the creation of rainbow tables for specific logins such as "root" or "administrator". All in all, as NIST recommends in [TBBC10], salts have to be unpredictable and to contain at least 128 randomly generated bits. This constraint is implicitly enforced by our definition (if the security parameter is set to 128 bits).

Password Hashing Scheme. It is defined by six 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 param;

- $\mathsf{PSalt}(\mathsf{param})$ is a probabilistic algorithm which takes as input the parameters param and outputs a salt s in some set $\mathbb{S}$ (depending on 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 param and a password $\pi \in \{0,1\}^n$, and outputs a pre-hash value P;
- $\mathsf{PHash}(\mathsf{param}, s, \pi)$ is a deterministic algorithm which takes as input the parameters param, a salt s and a password π, and outputs a hash value H;
- $\mathsf{PTSalt}(\mathsf{param})$ is a probabilistic algorithm which takes as input the parameters param and outputs a salt s in the set $\mathbb{S}$ (depending on param), together with a trapdoor ts corresponding to s;
- $\mathsf{PTCheck}(\mathsf{param}, s, \mathsf{ts}, P, H)$ is a deterministic algorithm which takes as input the parameters param, a salt s, a trapdoor ts (associated to s), a pre-hash value P, and a hash value, and outputs 1 if there exists a password π such that $\mathsf{PPreHash}(\mathsf{param}, \pi) = P$ and $\mathsf{PHash}(\mathsf{param}, s, \pi) = H$.

For convenience, we will sometimes omit param as argument.

The $\mathsf{PPreHash}$, PTSalt and $\mathsf{PTCheck}$ algorithms might look artificial, but they will help to move the algebraic part of the hashing in the PHash algorithm. This will allow the use of SPHFs, as in the GL framework. More precisely, thanks to $\mathsf{PPreHash}$, proving that a commitment or an encryption of a pre-hash value corresponds to some hash value, can be done "algebraically", as it is needed in the GL framework. This would not be the case, for example, if the pre-hash was just the password and the hash is SHA-256 of the password and the salt. 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. In a lot of cases, the hash value can be computed directly from the pre-hash value P (hence the name pre-hash), ts is not used and $\mathsf{PTCheck}(\mathsf{param}, s, \mathsf{ts}, P, H)$ just computes the hash value from P and s and check if it is H.

Basically, we then want the tight one-wayness property to ensure that given a set of hash values, it is hard to find a pre-hash value of any of the corresponding passwords, i.e., a pre-hash value passing the check $\mathsf{PTCheck}$. This is slightly stronger than finding a real password.

Remark 1. We chose to forbid $\mathsf{PPreHash}$ to use the salt s. Otherwise, since the salt s is not memorizable by the client, this would require the server to send the salt to the client at the beginning of our generic VPAKE protocol in Section 4.3, in an additional flow. In practice, since a 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.

Remark 2. We could extend our definition to handle randomized pre-hashing and hashing algorithms $\mathsf{PPreHash}$ and PHash, by adding other algorithms aiming at checking that some hash value corresponds to a given password. We do not do it to keep the model simple, as our constructions do not need it. We note that is somehow what Kiefer and Manulis did in [KM14].

Remark 3. Our definition supposes passwords are n-bit-strings. This is often not the case in practice, if passwords are ASCII strings containing only alphanumeric characters. 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. Note that we will not require uniform distribution, but use the min-entropy of the passwords.

Security Properties. In addition to correctness, in order to be applied in VPAKE, a password hashing scheme has to satisfy four security properties (all of them, except the last one, are statistical):

- *Correctness.* For any parameter $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}, 1^n)$, any password π, if $(s, \mathsf{ts}) \xleftarrow{\$} \mathsf{PTSalt}(\mathsf{param})$, $P \leftarrow \mathsf{PPreHash}(\mathsf{param}, \pi)$, and $H \leftarrow \mathsf{PHash}(\mathsf{param}, s, \pi)$, then $\mathsf{PTCheck}(\mathsf{param}, s, \mathsf{ts}, P, H) = 1$;
- *Salt indistinguishability.* For any parameter $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}, 1^n)$, the distribution of s, when $s \xleftarrow{\$} \mathsf{PSalt}(\mathsf{param})$ is the same as the distribution of s, when $(s, \mathsf{ts}) \xleftarrow{\$} \mathsf{PTSalt}(\mathsf{param})$;

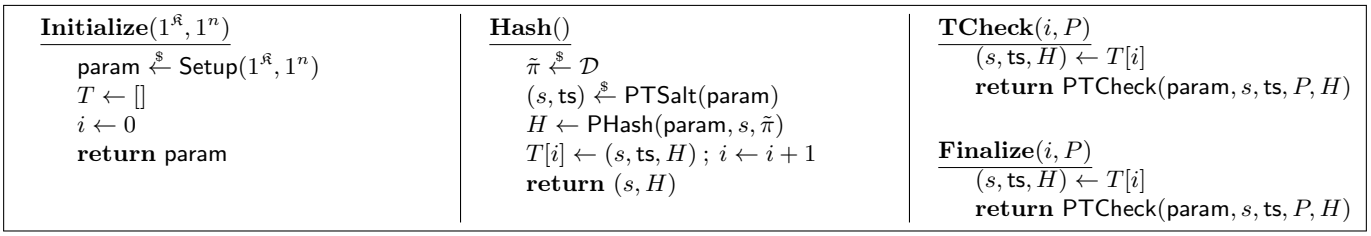

Fig. 1. Game $\mathbf{G}_{\texttt{one-way}}$ for the one-wayness of password hashing schemes

– *Second pre-image resistance.* There exists a negligible function[2] ε in $\mathfrak{K}$ such that, for any password $\pi \in \{0,1\}^n$, and any (unbounded) adversary $\mathcal{A}$:

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

– *Entropy preservation.* There exists a negligible function ε in $\mathfrak{K}$ such that, for any password distribution $\mathcal{D}$ of min-entropy β (not necessarily samplable in polynomial time), and any (unbounded) adversary $\mathcal{A}$:

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

– *Pre-hash entropy preservation.* There exists a negligible function ε in $\mathfrak{K}$ such that, for any password distribution $\mathcal{D}$ of min-entropy β (not necessarily samplable in polynomial time), and any adversary $\mathcal{A}$ running in polynomial time in $\mathfrak{K}$:

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

– *Tight one-wayness.* "Inverting" PHash (by inverting, we mean recovering a valid pre-hash value) takes approximately as long as evaluating 2^β times the function PHash, on average (i.e., brute-forcing the password), up to some small multiplicative constant, where β denotes the min-entropy of the password distribution $\mathcal{D}$. More precisely, let us consider the game $\mathbf{G}_{\texttt{one-way}}$ 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. He also has access to an oracle **TCheck** to check whether he guessed correctly the pre-hash value of some hash value. 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 hash values (and the corresponding salt) output by **Hash**. If the guess is correct, he wins. Notice that **TCheck** is similar to **Finalize**, except the latter ends the game.
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 β:

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

where t_{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, in the latter security game, 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. In practice, this means that the

[2] We do not use the notation negl here and in the two following properties, as we do want to insist on the fact that the same bound has to hold for all passwords and all distributions $\mathcal{D}$ for the other properties. In other words, we want the bound to be "uniform".

adversary just wants to find one password among a huge list of verifiers, and not only a specific one. This is enough for him to enter into the system and to make huge damages. This is the reason why salts are required: without salt, PHash evaluation would then target all the hash values.

The first three security properties are actually (at least implicitly) expected for all hash functions, and are always straightforward in our case. Intuitively, they ensure that, if a password π is chosen according to some password distribution $\mathcal{D}$ of min-entropy β, then the adversary cannot find the hash value of π (entropy preservation), or a pre-hash value corresponding to the hash value of π (second pre-image resistance and pre-hash entropy preservation) with probability significantly more than $2^{-\beta}$. The technical part will be to design a scheme that satisfies the tight one-wayness, and to prove it! In addition, for our best protocols, we will need password hashing schemes that admit SPHFs, to be compatible with the GL framework, the most efficient known so far without random oracles. Let us remind that avoiding idealized models is our main goal. This will require PHash to satisfy some algebraic properties.

Idealized Models. Due to the strict constraints on running time of adversaries in the tight one-wayness definition, it seems really hard, if not impossible, to prove it under "classical" assumptions, such as DDH. That is why we make use of idealized models, where the running time of the adversary is replaced by the number of queries to some oracle.

For example, in the random oracle model, instead of considering the running time t of the adversary and the time t_{PHash} of a call to PHash, we consider $q_{\mathcal{H}}$ the number of adversarial queries to the random oracle and q_{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-256, 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.

We point out that, as explained in Section 1.1, the generic group model and the random oracle model are only used to prove the tight one-wayness of our password hashing schemes, or more precisely, to show that the tight one-wayness of our schemes are reasonable assumptions. For our VPAKE construction, we will never use directly these idealized models, but only suppose that the underlying password hashing scheme is tightly one-way.

2.3 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, $\mathsf{param} = 1^{\mathfrak{K}}$, $\mathsf{PSalt}(\mathsf{param})$ and $\mathsf{PTSalt}(\mathsf{param})$ output a random salt $s \in \mathbb{S} = \{0,1\}^{2\mathfrak{K}}$ ($\mathsf{ts} = \perp$), $P = \mathsf{PPreHash}(\pi) = \pi$, $H = \mathsf{PHash}(s,\pi) = \mathcal{H}(s,\pi)$, and $\mathsf{PTCheck}(s,\mathsf{ts},P,H)$ just check whether $\mathcal{H}(s,P) = H$. Let us show that this scheme is a password hashing scheme for any $n \leq \mathfrak{K}$:

- *Correctness* and *salt indistinguishability* are trivial.
- *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,π) has been asked by the adversary to the random oracle, in which case $\mathcal{H}(s,\pi) \neq H$, or (s,π) 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 procedure **TCheck** can be easily simulated and is not useful. Then, 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 seems hard to be used in a VPAKE scheme, or at least, we will not be able to use it within our GL approach (see Section 4.3), because PHash has no algebraic property and so seems not compatible with any SPHF. Let us now introduce another construction which is used later in one of our VPAKE schemes. Let $\mathsf{param} = (\mathbb{G}, g)$ with $\mathbb{G}$ a (multiplicative) cyclic group of order p (a prime number with more than $2\mathfrak{K}$ bits) and g a generator of $\mathbb{G}$, and let $\mathcal{H}$ 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 $P = \mathsf{PPreHash}(\pi) = g^{\mathcal{H}(\pi)}$ and $H = \mathsf{PHash}(a, \pi) = a^{\mathcal{H}(\pi)}$. Finally, $\mathsf{PTSalt}(\mathsf{param})$ picks a random scalar $\mathsf{ts} \in \mathbb{Z}_p^*$ and outputs $(s = a = g^{\mathsf{ts}}, \mathsf{ts})$, and $\mathsf{PTCheck}(s, \mathsf{ts}, P, H)$ checks that $P^{\mathsf{ts}} = H$.

We could have chosen a simpler construction where $P = \mathcal{H}(\pi)$. In that case, ts could have been $\perp$. But this would have yield less efficient VPAKE construction, because we would need to encrypt P bit-by-bit.

Let us now prove the security of the construction:
– *Correctness* and *salt indistinguishability* are straightforward.
– *Second pre-image resistance.* For any passwords $\pi, \pi' \in \{0,1\}^n$ and any salt $s = a \in \mathbb{S}$, if $\mathsf{PHash}(a, \pi) = \mathsf{PHash}(a, \pi')$, $a^{\mathcal{H}(\pi)} = a^{\mathcal{H}(\pi')}$ and so $\mathcal{H}(\pi) = \mathcal{H}(\pi')$, since a is a generator of order p. Hence $\mathsf{PPreHash}(\pi) = \mathsf{PPreHash}(\pi') = g^{\mathcal{H}(\pi)}$, and 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 μ arbitrary passwords $\pi'_1, \ldots, \pi'_\mu \in \{0,1\}^n$ to get μ values $\mathcal{H}(\pi'_1) = \hat{\pi}'_1, \ldots, \mathcal{H}(\pi'_\mu) = \hat{\pi}'_\mu \in \mathbb{Z}_p$ and do μ requests to **Hash** and gets μ salts $s_1, \ldots, s_\mu$ and μ hash values $H_1 = \mathsf{PHash}(s_1, \pi_1), \ldots, H_\mu = \mathsf{PHash}(s_\mu, pw_\mu)$. 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, $\mathsf{PHash}(s_i, \pi'_j) = H_i$. 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 $\mathsf{PHash}(s_i, \pi_j)$, and thus 2^n exponentiations.
In Appendix C.1, we prove that any adversary has to do at least about 2^β operations in the generic group model, for any n such that $2^{4n+1} < p$ (i.e., $n \lesssim \mathfrak{K}/2$), where β is the min-entropy of the distribution of the passwords. This proves the tight one-wayness since the cost of evaluating PHash 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 Appendix C.1 because the original proof was not rigorous enough for our purpose.

Remark 4 (on the use of the random oracle model). One may wonder whether it is possible to prove the tight one-wayness, if $\mathcal{H}$ is just collision-resistant (and not modeled as a random oracle), since the random oracle is not programmed here and is just used to ensure that the pre-hash values look random. Unfortunately, it is impossible. For example, if we choose $\mathcal{H}$ to be the identity function (where passwords in $\{0,1\}^n$ are seen as integers in $\{0, \ldots, 2^n - 1\}$), then given a salt s and the hash value $s^{\mathcal{H}(\pi)} = s^\pi$ of π, we can recover π in time $O(2^{n/2})$, using a variant of the Pollard's kangaroo method described in [MT09]. Since the uniform distribution $\mathcal{D}$ over $\{0,1\}^n$ has min-entropy n, this means the tight one-wayness does not hold.

Remark 5 (on the use of the generic group model). We could avoid using the generic group model, and simply rely on the DDH assumption in the random oracle model, if we would allow to use a salt $s \in \{0,1\}^{2\mathfrak{K}}$ in

PPreHash, by setting $P = g^{\mathcal{H}(s,\pi)}$, and $H = h^{\mathcal{H}(s,\pi)}$ for some other generator h. But as already explained, this is not supported by our password hashing schemes, since it would require the server to send the salt to the client at the beginning of our generic VPAKE protocol in Section 4.3, in an additional flow. Therefore we preferred to forbid the use of a salt in PPreHash at the cost of relying on the generic model for this password hashing scheme. Anyway, this variant where P depends on the salt would not yield a more efficient VPAKE using our generic construction in Section 4 (and actually, it not only adds one round, but also require to add a one-time signature to partially authenticate the flows, due to this additional round).

In Section 5, we present another password hashing scheme, that will make use of multilinear maps [GGH13]. While in practice this second password hashing scheme is way less efficient that our first hashing scheme, it is interesting in theory as its proof of tight one-wayness only relies on the generic model, but not on the random oracle model. Before describing it, we first provide our extension of the BPR model to VPAKE protocols, then we describe our generic VPAKE protocol with the security analysis. Our SPHF-friendly password hashing schemes will immediately fit within this framework.

3 Security Models

In this section, we introduce our new game-based security model for VPAKE protocols, associated with a password hashing scheme. This is 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.

3.1 A BPR-like Security Model

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

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 π_C, while each server $S \in \mathcal{S}$ holds a random salt $s_S \xleftarrow{\$} \mathsf{PSalt}(\mathsf{param})$ and a hash value $H_S = \mathsf{PHash}(\mathsf{param}, s_S, \pi_S)$ of some password π_S. Without lost of generality, we suppose that each client C is paired with a server S, and that for each matching pair (C, S), $\pi_C = \pi_S$. For the sake of simplicity, each client or server appears exactly in one pair. We point out that this is not a restriction of the model, and that it is possible to execute the protocol between a client C and a server S which are not matching.

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:

- $\texttt{Execute}(C^i, S^j)$: 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 π_S is said to have his password hash corrupted.

Since our goal is to limit damages in case of server corruption, we do not consider client corruption queries, and thus do not deal with forward secrecy, but we discuss it later.

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, the output is $\perp$;
- if U^i is a partnered client instance whose password hash has been corrupted, the output is $\perp$ (as the adversary can trivially simulate the server in this case);
- if a `Test` query has already been ask to U^i, the output is the same as for the previous query;
- if a `Test` query has been ask to U^i's partner (if he has any), then if they are matching client-server, the output is the same as for the previous partner's query, whereas if they are not matching, the output is a random session key;
- 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 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), the `Test` query is answered by real values or previously sent values.

A VPAKE could be considered secure if the advantage of any adversary $\mathcal{A}$, running in time t, in the previous experiment is upper-bounded by:
- $q_s \times 2^{-\beta} + \mathrm{negl}(\mathfrak{K})$, if the adversary did not ask a `Test`-query for a server with a corrupted password hash, and who received at least one flow generated by the adversary;
- $q_s \times 2^{-\beta} + \mathsf{Adv}_{\texttt{one-way}}(\mathcal{B}, \mathfrak{K}) + \mathrm{negl}(\mathfrak{K})$, 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) and passwords of each matching pair (C, S) is chosen independently according to some distribution $\mathcal{D}$ (samplable in polynomial time) of min-entropy β. 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 $q_s \times 2^{-\beta}$ in both bounds).

Remark. In the BPR model [BPR00], there was an additional `Reveal`-query which outputs the actual session (as the `Test`-query does in the real and trivial cases). But it has been proven in [AFP05] that multi-`Test`-queries with or without `Reveal`-queries are equivalent.

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 satisfied by the GL construction.

3.2 Related-Password Model and Forward-Secrecy

Note that the main differences of the BPR-model [BPR00] with the UC-model [CHK+05] are possible relations between passwords with non uniform distributions and the forward-secrecy.

The latter can easily be handled by adding another query: $\texttt{Corrupt}(C)$, which models the client corruption. The output of this query is the password π_C. The aim of forward secrecy is that even knowing a password should not help to distinguish previous session keys from random keys. As a consequence, for corrupted

clients and their matching partners, `Test`-queries are answered with the actual session keys (but only after the client corruption, since keys established before should remain random-looking). We do not modify `Test`-query answers in case of server corruption since we are still dealing with VPAKE schemes. Applying the same modification would annihilate the role of the password hashing.

More concretely, the client corruption query allows to show that all the previous keys remain secure even in case of the password-corruption, and the server corruption query allows to express the quality of the password hashing scheme.

For related and non-uniform passwords, instead of supposing that passwords of each pair (C, S) are chosen independently according to some distribution $\mathcal{D}$. We now choose all passwords from a complex joint distribution $\mathcal{D}$ over $(\{0,1\}^n)^Q$ (samplable in polynomial time), where Q is a bound on the number of possible matching pairs (C, S).

We remark that we cannot set β to be the minimum min-entropy of the password of one client C_i, according to $\mathcal{D}$, because it is possible that the password π_1 of C_1 is equal to the password π_2 of C_2, in which case the adversary could corrupt C_1, get $\pi_1 = \pi_2$ and use it to win the game. More subtly, we recall that the adversary can make two non-matching players (C, S) to play together and know whether they used the same password or not. Therefore, we could imagine a distribution $\mathcal{D}$ where the i-th bit of the password π_0 of client C_0 is 1 if and only is the password π_{2i} of C_{2i} is the same as the password π_{2i+1} of C_{2i+1}. Then, just by executing protocols between C_{2i} and S_{2i+1} (assuming matching pairs are of the form (C_j, S_j)), the adversary can get all the bits of π_0 and win the game. These attacks are unavoidable.

That is why, in our related-password model, we need to take into account, not the entropy β of each password π, but the remaining min-entropy after all the corruption queries and the equality queries. An equality query between two passwords π_i and π_j is performed by the adversary when he makes a `Test`-query on a client instance $C_i^{i'}$ (with password π_i) and a server instance $S_j^{j'}$ which are partnered and between which the protocol was executed honestly (either directly with an `Execute` query, or via `Send` queries without alteration of messages).

We say that an adversary $\mathcal{A}$ preserves a min-entropy of β, when for any random tape r of $\mathcal{A}$:

$$-\log_2 \max_{(\pi_C)_C \in (\{0,1\}^n)^Q} \Pr_{(\pi'_C)_C \xleftarrow{\$} \mathcal{D}} \left[(\pi'_C) = (\pi_C) \;\middle|\; \begin{array}{l} \text{corruption and equality queries of } \mathcal{A} \text{ with random tape} \\ r \text{ are the same and give the same results when pass-} \\ \text{words are } (\pi'_C) \text{ and when passwords are } (\pi_C) \end{array} \right] \geq \beta$$

We then say that a VPAKE protocol is secure even with related passwords, and provides forward-secrecy, if the advantage of any adversary $\mathcal{A}$, running in time t, and preserving min-entropy of β, is upper-bounded by:

- $q_s \times 2^{-\beta} + \mathrm{negl}(\mathfrak{K})$, if the adversary did not ask a `Test`-query to a server with a corrupted password hash, and who received at least one flow generated by the adversary;
- $q_s \times 2^{-\beta} + \mathsf{Adv}^{\texttt{one-way}}(\mathcal{B}, \mathfrak{K}) + \mathrm{negl}(\mathfrak{K})$, otherwise, for some adversary $\mathcal{B}$ running in time about at most t.

PAKE: as above, using the trivial password hashing scheme, we get a security model for PAKE that handles related passwords. This stronger model is satisfied by a variant[3] of the GL construction, as shown in Section 4.3.

4 VPAKE Constructions

Since the GL framework, with SPHFs, is the unique approach to build efficient PAKE protocols secure in the standard model, we follow this path to propose generic VPAKE constructions using SPHFs.

4.1 Tools

Smooth Projective Hash Functions. Projective hash function families were first introduced by Cramer and Shoup [CS02]. Here we use the formalization from [BBC+13c]: 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 hk or a *public* projection key hp but with a witness w of the fact that c is indeed in L:

[3] Actually, we could prove the security of the original GL construction (with related passwords) by making slight changes in our proof.

– $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L;
– $\mathsf{ProjKG}(\mathsf{hk}, L, c)$ derives the projection key 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 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 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 hp), we say the SPHF is a KVSPHF. Otherwise, it is said to be a GLSPHF. See [BBC+13c] 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 pk and a secret key sk;
– $\mathsf{Enc}^{\ell}(\mathsf{pk}, x; r)$ encrypts the message x under the key pk and the label ℓ, and using the random coins r;
– $\mathsf{Dec}^{\ell}(\mathsf{sk}, c)$ decrypts the ciphertext c with the label ℓ using the secret key sk.

In this paper, we consider `IND-CPA` (without label) and `IND-CCA` (with labels) encryption schemes [BDPR98].

4.2 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 π_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 $H_C = \mathsf{PHash}(\mathsf{param}, s_S, \pi_C)$. 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 such that there exists π_C satisfying $\mathsf{PPreHash}(\mathsf{param}, \pi_C) = P_C$ and $\mathsf{PHash}(\mathsf{param}, s_S, \pi_C) = H_S$.

Link with LAKE. Our construction can roughly be seen as a particular case of a LAKE (Language Authenticated Key Exchange) [BBC+13a], which is an extension of a PAKE where each player knows a private information (like a password), a word, and a language. The key exchange is successful if private information is the same and if the word of each player is in the language of the other player. In our case, the private information is the hash value $H_S = H_C$, and the server does not hold any word and the client holds a word $P_C = \mathsf{PPreHash}(\mathsf{param}, \pi_C) \in \{P \mid \exists \pi \in \{0,1\}^n, \mathsf{PPreHash}(\mathsf{param}, \pi) = P \text{ and } \mathsf{PHash}(\mathsf{param}, s_S, \pi) = H_S\}$.

However, if we actually base our construction on existing LAKE constructions such as [BBC+13a,BBC+13c, ABB+13], we cannot directly compose these LAKEs with a password hashing scheme, because these constructions have not been 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+13c] 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 account in these previous models[4].

However, it seems that using any of the constructions mentioned above should work. Since in this paper, we are not interested in UC-secure schemes, our generic VPAKE scheme in Section 4.3 is based on a two-round variant of the LAKE construction in [BBC+13c], using a KVSPHF and a GLSPHF instead of two KVSPHFs.

[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 uses the same salt, between two executions of the protocol, since the client only knows the user's password.

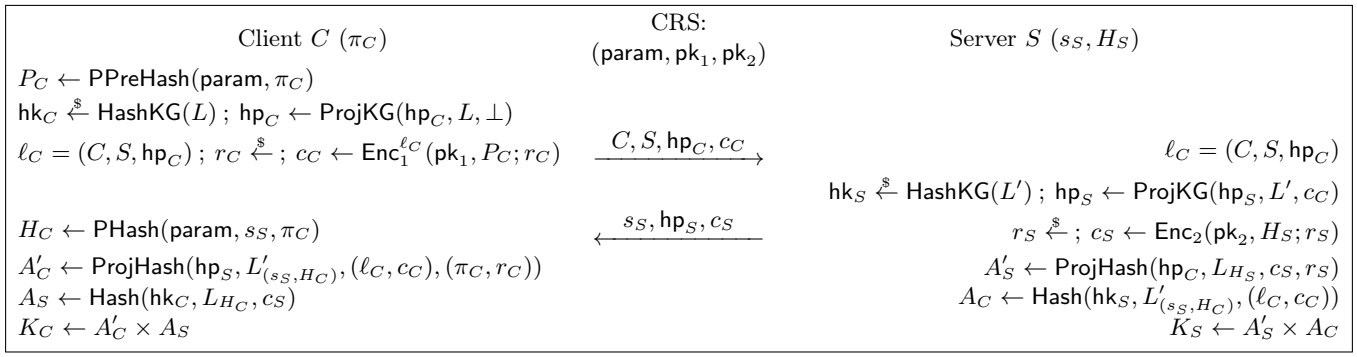

Fig. 2. Generic two-round VPAKE Construction

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.

4.3 Generic Two-Round Construction

Construction. Let us consider a labeled IND-CCA encryption scheme $(\mathsf{KG}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$, an IND-CPA one $(\mathsf{KG}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$, and a password hashing scheme $\mathsf{PH} = (\mathsf{Setup}, \mathsf{PSalt}, \mathsf{PPreHash}, \mathsf{PHash})$. Let us suppose we have a GLSPHF and a KVSPHF (respectively) for the two following families of languages:

$$L'_{(s,H)} = \{(\ell, c) \mid \exists \pi, \exists r,\ c = \mathsf{Enc}_1^{\ell}(\mathsf{pk}_1, \mathsf{PPreHash}(\mathsf{param}, \pi); r) \text{ and } H = \mathsf{PHash}(\mathsf{param}, s, \pi)\}$$
$$L_H = \{c \mid \exists r,\ c = \mathsf{Enc}_2^{\ell}(\mathsf{pk}_2, H; r)\},$$

with param, pk_1 and pk_2 global parameters in the common reference string CRS, generated as follows: $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}, 1^n)$, $(\mathsf{sk}_1, \mathsf{pk}_1) \xleftarrow{\$} \mathsf{KG}_1(1^{\mathfrak{K}})$ and $(\mathsf{sk}_2, \mathsf{pk}_2) \xleftarrow{\$} \mathsf{KG}_2(1^{\mathfrak{K}})$; and with $s \in \mathbb{S}$ and H a hash value of some password π. We also suppose the HashKG and ProjKG for $L'_{(s,H)}$ and L_H do not depend on H. In other words, they just work for $L' = \{(\ell, c) \mid \exists r, \exists P,\ c = \mathsf{Enc}_1^{\ell}(\mathsf{pk}_1, P; r)\} \supset L'_{(s,H)}$ (for any s, H) and $L = \{c \mid \exists r, \exists H,\ c = \mathsf{Enc}_2(\mathsf{pk}_2, H; r)\} \supset L_H$. 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 (independent of B), $A \times B$ is uniform (often hash values live in a group and $\times$ is just the group law).

This scheme is secure in our BPR-like model for VPAKE, with forward-secrecy, related passwords, and static corruptions (where the adversary is only allowed to corrupt a client C or a server S of some matching pair (C, S) when no instance of C nor S is involved in an execution of the protocol). Achieving security against static corruptions is already quite challenging [GK10]. This security proof is similar to the one in [BBC+13c] and can be found in Appendix C.3.

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 encryption schemes have to be IND-CCA.

PAKE Variants. If we consider the previous model with the trivial (insecure) password hashing scheme, where salts are $\bot$ and $\mathsf{PHash}(\mathsf{param}, \bot, \pi) = \pi$, the one-round generic protocol is exactly the one of Katz and Vaikuntanathan [KV11], while the two-round generic protocol is a folklore (improved) variant of the GL framework [GL06] (where one-time signatures are no more required since the protocol is 2-round instead of 3-round, and the server encryption scheme is only IND-CPA instead of IND-CCA).

It is worth mentionning that our proof also shows that these constructions are secure with related passwords, which was not known, up to now.

[5] Namely, $\ell_C = (C, S, \mathsf{hp}_C)$ and $\ell_S = (C, S, \mathsf{hp}_S)$, with two IND-CCA labeled-encryption schemes.

4.4 Instantiations and Comparison

In this section we briefly show how to instantiate this generic scheme with our password hashing schemes. It indeed applies to the algebraic password hashing scheme presented in Section 2.3, but also to the one we introduce in the next section that makes use of multilinear maps, without random oracles. To instantiate our generic scheme, we just need to choose IND-CPA and IND-CCA encryption schemes and to construct SPHFs for the languages L_H and $L'_{(s,H)}$. Details can be found in Appendix B, due to lack of space.

Instantiation with our Algebraic Password Hashing Scheme of Section 2.3. For the two-round version, we use the labeled Cramer-Shoup encryption scheme for Enc_1 and ElGamal for Enc_2, while for the one-round version, we use the labeled Cramer-Shoup encryption scheme for Enc_1 and Enc_2. One the one hand, in the two-round version, the client sends 5 group elements (4 for c_C and 1 for hp_C) and the server sends 5 groups elements (2 for c_S, 1 for the salt s_S, and 2 for hp_S). This is a total of 10 groups elements, just three more than the most efficient PAKE in the BPR model [GK10]. On the other hand, in the one-round version, the client sends 6 group elements (4 for c_C and 2 for hp_C) and the server sends 8 group elements (4 for c_S, 1 for the salt s_C, and 3 for hp_S). This is a total of 12 group elements, just two more than the most efficient one-round PAKE in the BPR model [BBC$^+$13c]. So even compared to PAKE, our VPAKE is very competitive.

Compared to the other VPAKE constructions in the standard model (assuming the tight one-wayness of the underlying password hashing scheme), our construction outperforms both in term of rounds, namely 1 or 2 (compared to at least 3), and in term of communication complexity: some previous such constructions consist in a classical PAKE with the verifier followed by a zero-knowledge proof (e.g., Schnorr-like as in [ACP05]) that the client knows the password associated to the verifier; some other constructions just require a much higher communication complexity and a high (≥ 3) number of rounds by design (e.g., [CCGS10] and [BBC$^+$13a]). Last but not least, our protocols are formally proven secure within a new security model, that is the first one allowing provably secure protocols in the standard model.

Instantiation with our Password Hashing with Multilinear Maps of Section 5.2. We only propose the two-round protocol since no efficient KVSPHF seem to exist. The encryption scheme Enc_1 is the labeled Cramer-Shoup encryption scheme for vectors as recalled in Appendix B.1, and the encryption scheme Enc_2 is the ElGamal encryption scheme.

The security of all our instantiations directly comes from the security of our password hashing schemes (proven in the generic group model and in the generic multilinear 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 assumptions, our instantiations would be secure in the standard model.

5 Multilinear Maps

5.1 Multilinear Maps and Graded Rings

In the following, we use an idealized version of asymmetric multilinear maps with the notations introduced in [BBC$^+$13b]. 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 and a discussion on their implementation can be found in Appendix A.1.

5.2 Construction of a Password Hashing System

Let us now introduce a completely algebraic construction based on multilinear 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 $H = a^\pi$ (where $\pi \in \{0,1\}^n$ is seen as an integer in $\{0, \ldots, 2^n - 1\}$). But, as explained in Remark 4, a baby-step giant-step algorithm or a variant of the Pollard's kangaroo method described in [MT09] enables 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 Multilinear Maps. Let us now consider an $(n+1)$-graded ring $\mathfrak{G}$, and let $(a_{i,b})_{i,b}$ (with $1 \leq i \leq n$ and $b \in \{0,1\}$) be random non-zero 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-zero group element $s \in \mathbb{S} = \mathfrak{G}_{\boldsymbol{e}_{n+1}} \setminus \{0\}$, while $P = \mathsf{PPreHash}(\pi) = \pi$ and $H = \mathsf{PHash}(s, \pi) = a_{1,\pi[1]} \odot \cdots \odot a_{n,\pi[n]} \odot s$. Since $P = \pi$, no trapdoor ts is needed, and $\mathsf{PTCheck}(s, \bot, P, H)$ just checks whether $\mathsf{PHash}(s, \pi) = H$. 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. However the security proof does not follow from [BR13] as their proof is not tight as we understand it. Our proof is in the generic graded ring 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, the CFM Foundation, and the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

References

ABB+13. M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, and D. Pointcheval. SPHF-friendly non-interactive commitments. In *ASIACRYPT 2013, Part I*, *LNCS* 8269, pages 214–234. Springer, December 2013. (Pages 3 and 12.)

ACP05. M. Abdalla, O. Chevassut, and D. Pointcheval. One-time verifier-based encrypted key exchange. In *PKC 2005*, *LNCS* 3386, pages 47–64. Springer, January 2005. (Page 14.)

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 3.)

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 3, 9, and 10.)

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 3.)

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 12 and 14.)

BBC+13b. 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/2013/034`. (Page 14.)

BBC+13c. 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, 19, and 20.)

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 18.)

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 3.)

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 3.)

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 3.)

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 12.)

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 3.)

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 3, 9, and 10.)

BR13. Z. Brakerski and G. N. Rothblum. Obfuscating conjunctions. In *CRYPTO 2013, Part II*, *LNCS* 8043, pages 416–434. Springer, August 2013. (Pages 3, 15, 18, 26, and 27.)

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 3.)

CCGS10. J. Camenisch, N. Casati, T. Groß, and V. Shoup. Credential authenticated identification and key exchange. In *CRYPTO 2010*, *LNCS* 6223, pages 255–276. Springer, August 2010. (Page 14.)

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. (Pages 3 and 10.)

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. (Page 18.)

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 3, 11, and 19.)

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 9, 16, and 18.)

GK10. A. Groce and J. Katz. A new framework for efficient password-based authenticated key exchange. In *ACM CCS 10*, pages 516–525. ACM Press, October 2010. (Pages 2, 13, and 14.)

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 3 and 28.)

GL06. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. *ACM Transactions on Information and System Security*, 9(2):181–234, 2006. (Pages 2, 3, and 13.)

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. (Pages 2 and 3.)

KM14. F. Kiefer and M. Manulis. Zero-knowledge password policy checks and verifier-based PAKE. In *ESORICS 2014, Part II*, *LNCS* 8713, pages 295–312. Springer, September 2014. (Pages 3, 4, and 5.)

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 3.)

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, 3, and 13.)

MMN06. I. Mironov, A. Mityagin, and K. Nissim. Hard instances of the constrained discrete logarithm problem. In *Algorithmic Number Theory*, pages 582–598. Springer, 2006. (Page 21.)

MT09. R. Montenegro and P. Tetali. How long does it take to catch a wild kangaroo? In *41st ACM STOC*, pages 553–560. ACM Press, May / June 2009. (Pages 4, 8, and 15.)

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 8 and 21.)

TBBC10. M. S. Turan, E. Barker, W. Burr, and L. Chen. Recommendation for password-based key derivation — part 1: Storage applications, 2010. NIST Special Publication 800-132. (Page 4.)

A Graded Rings

A.1 Graded Rings

Let us first recall the notion of graded ring introduced in [BBC+13c], 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 multilinear 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 Λ, we also consider Λ 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 Λ ($\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$).

[6] In the original definition in [BBC+13c], the authors consider more general sets Λ.

Graded Ring. The τ-graded ring over a commutative ring R is the set $\mathfrak{G} = \Lambda \times R = \{[\boldsymbol{v}, x] \,|\, \boldsymbol{v} \in \Lambda, x \in R\}$, 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 $\{u = [\boldsymbol{v}', x] \in \mathfrak{G} \,|\, \boldsymbol{v}' = \boldsymbol{v}\}$ 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:

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

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 6 (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 7 (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 8 (Graded External DDH Assumption (GXDH)). *Let $\mathfrak{G}$ be a τ-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 operation.

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 4.4, 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 some of our VPAKE constructions, we use a variant of the Cramer-Shoup encryption scheme for vectors of m messages, with randomness reuse [BBS03], when $m > 1$ (of course, when $m = 1$, this is the usual Cramer-Shoup). This scheme has already been used in [BBC+13c]. 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_{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 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 MCS, is IND-CCA under the 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). In addition, if IND-CPA is enough, we can use the ElGamal scheme (with randomness reuse in case of vector encryption), which drops v and w, we name it the Multi-ElGamal encryption scheme, denoted MEG.

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

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 IND-CCA secure as long as the 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 and one in one-round. The second one slightly less efficient but still very practical. Both instantiations work in a cyclic group $\mathbb{G}$ of order p.

Let $\mathsf{pk}_1 = (g_1, h_1, c_1, d_1, f_1)$ be a public key for the labeled Cramer-Shoup encryption scheme (of vector of size 1), let $\mathsf{pk}_2 = (g_2, f_2)$ be a public key for the ElGamal encryption scheme, and let $\mathsf{param} = (\mathbb{G}, g)$ (with g another generator of $\mathbb{G}$) be the parameters of the password hashing scheme in Section 2.3.

We could actually use $g_1 = g_2$ and $f_1 = f_2$, and the proof would still work. But for the sake of clarity, we do not do it here.

Two-Round Instantiation. For this instantiaion Enc_1 is the Cramer-Shoup encryption scheme, while Enc_2 is the ElGamal encryption scheme.

Using notations of the generic two-round scheme in Section 4.3, we then have:

$$\begin{aligned} c_C &= \mathsf{Enc}_1^{\ell}(\mathsf{pk}_1, P_C; r_C) \\ &= (u_C = g_1^{r_C}, v_C = h^{r_C}, e_C = f^{r_C} \cdot P_C, w_C = (cd^{\theta_C})^{r_C}), \end{aligned}$$

with $\theta_C = H_{\mathsf{CS}}(\ell_C, u_C, v_C, e_C)$, and $P_C = g^{\mathcal{H}(\pi_C)}$ the pre-hash value of the client password π_C; and

$$\begin{aligned} c_S &= \mathsf{Enc}_2(\mathsf{pk}, P; r_S) \\ &= (u_S = g^{r_S}, e_S = f^{r_S} \cdot H_S) \end{aligned}$$

with $H_S = s_S^{\mathcal{H}(\pi_S)}$ the hash value of the password π_S and s_S the salt.

It remains to construct a KVSPHF for

$$\begin{aligned} L'_{(s,H)} &= \{(\ell, c) \mid \exists r, \exists \pi,\ \mathsf{Enc}_1[\ell](\mathsf{pk}, \mathsf{PPreHash}(\mathsf{param}, \pi); r) \text{ and } \mathsf{PHash}(\mathsf{param}, s, \pi) = H\} \\ &= \{(\ell, c = (u, v, e, w)) \mid \exists r \in \mathbb{Z}_p, \exists \hat{\pi} \in \mathbb{Z}_p,\ u = g_1^r, v = h_1^r,\ e = f_1^r \cdot g^{\hat{\pi}},\ w = (cd^{\theta})^r \text{ and } s^{\hat{\pi}} = H\} \end{aligned}$$

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

$$\begin{aligned} L_H &= \{c \mid \exists r,\ \mathsf{Enc}_2(\mathsf{pk}, H; r) = c\} \\ &= \left\{ c = (u, e) \;\middle|\; \exists r \in \mathbb{Z}_p,\ u = g_2^r, e = h_2^r \cdot g_2^{\hat{\pi}} \right\}. \end{aligned}$$

KVSPHF for L_H. L_H is just the language of ElGamal ciphertext of some given value H. Such an SPHF was already proposed in [CS02]. Let us just recall how to write this SPHF in the generic framework introduced in [BBC+13c], as a warm-up for the following SPHFs.

We use the following matrices:

$$\Gamma = \begin{pmatrix} g & f \end{pmatrix} \qquad \begin{aligned} \boldsymbol{\lambda} &= r \\ \boldsymbol{\lambda} \cdot \Gamma &= (g_2^r, f_2^r) \\ \Theta(C) &= (u, e) \end{aligned}$$

With $\mathsf{hk} = (\eta, \beta) \xleftarrow{\$} \mathbb{Z}_p^5$, we get $\mathsf{hp} = (g_2^{\eta} h_2^{\beta}) \in \mathbb{G}^2$. 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 $L'_{(s,H)}$. For this SPHF, we use the following matrices:

$$\Gamma = \begin{pmatrix} g_1 & h_1 & f_1 & 1 & (c_1 d_1^{\theta}) \\ 1 & 1 & g_1 & s & 1 \end{pmatrix} \qquad \begin{aligned} \boldsymbol{\lambda} &= (r, \mathcal{H}(\pi)) \\ \boldsymbol{\lambda} \cdot \Gamma &= (g_1^r, h_1^r, f_1^r g_1^{\mathcal{H}(\pi)}, s^{\mathcal{H}(\pi)}, (cd^{\theta})^r) \\ \Theta(C) &= (u, v, e, H, w). \end{aligned}$$

With $\mathsf{hk} = (\eta, \nu, \alpha, \beta, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$, we get $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta} h_1^{\nu} f_1^{\alpha} (c_1 d_1^{\theta})^{\mu}, \mathsf{hp}_2 = g_1^{\alpha} s^{\beta}) \in \mathbb{G}^2$.

One-Round Instantiation. This is the same as the two-round instantiation except the encryption scheme Enc_2 for the server is also IND-CCA and is labeled, and the two SPHFs have to be KVSPHF. Let us just construct the two SPHFs we need.

KVSPHF for L_H. L_H is just the language of Cramer-Shoup ciphertext of some given value H. In [BBC+13c], the authors already introduced a KVSPHF for this exact language. The projection key hp contains two group elements.

KVSPHF for $L'_{(s,H)}$. Notice we cannot use the same SPHF as for the two-round instantiation because it is only a GLSPHF (hp depends on θ which depends on the ciphertext c). For this SPHF, we thus use the following matrices:

$$\Gamma = \begin{pmatrix} g_1 & 1 & h_1 & f_1 & 1 & c_1 \\ 1 & g_1 & 1 & 1 & 1 & d_1 \\ 1 & 1 & 1 & g_1 & s_1 & 1 \end{pmatrix} \qquad \begin{aligned} \boldsymbol{\lambda} &= (r, r\theta, \mathcal{H}(\pi)) \\ \boldsymbol{\lambda}\cdot\Gamma &= (g_1^r, g_1^{r\theta}, h_1^r, f_1^r g_1^{\mathcal{H}(\pi)}, s_1^{\mathcal{H}(\pi)}, (cd^\theta)^r) \\ \Theta(C) &= (u, u^\theta, v, e, H, w). \end{aligned}$$

With $\mathsf{hk} = (\eta_1, \eta_2, \nu, \alpha, \beta, \mu) \xleftarrow{\$} \mathbb{Z}_p^6$, we get $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} h_1^\nu f_1^\alpha c_1^\mu, \mathsf{hp}_2 = g_1^{\eta_2} d_1^\mu, \mathsf{hp}_3 = g_1^\alpha s^\beta) \in \mathbb{G}^3$.

B.3 Instantiation Based on Multilinear 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 π_C.

Let $\mathsf{param} = (\mathfrak{G}, (a_{i,b})_{i,b})$ be the parameters of the password hashing scheme in Section 2.3 (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 4.3, we then have:

$$\begin{aligned} c_S = \mathsf{Enc}_2(\mathsf{pk}_0, H_S; r_S) &= 1\text{-}\mathsf{MEG}_{\mathsf{pk}_\top}((H_S); r_S) \\ &= (u_S = r_S \odot g_\top, e_S = r_S \odot f_\top \oplus H_S \odot g_0), \end{aligned}$$

and:

$$\begin{aligned} c_C = \mathsf{Enc}_1^{\ell_C}(\mathsf{pk}_1, \pi_C; r_C) &= |p|\text{-}\mathsf{MCS}_{\mathsf{pk}_0}^{\ell_C}((\pi_C[i] \odot g_0)_i; r_C) \\ &= (u_C = r_C \odot g_0, v_C = r_C \odot h_0, (e_{C,i} = r_C \odot f_{1,i} \oplus \pi_C[i] \odot g_0)_i, w_C = r_C \odot (c_0 \oplus (\theta_C \odot d_0))). \end{aligned}$$

with $\theta_C = H_{\mathsf{CS}}(\ell_C, u_C, v_C, (e_{C,i})_i)$. The dependance between the keys used in c_C and C_S is not a problem in the proof.

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 contains Cramer-Shoup encryption 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+13c]. The second part uses the matrices detailed on Figure 3.

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 $s \odot a_{1,x_1} \odot \cdots \odot a_{n,x_n} = H$.

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

$$\Theta(c) = (s \odot u,\ s \odot \epsilon_1,\ 0,\ 0,\ \ldots,\ 0,\ H \odot g_0)$$

$$\begin{aligned} \boldsymbol{\lambda} = (&r \odot s,\ s \odot a_{1,\pi[1]},\ r \odot s \odot a_{1,\pi[1]},\\ &s \odot a_{1,\pi[1]} \odot a_{2,\pi[2]},\ r \odot s \odot a_{1,\pi[1]} \odot a_{2,\pi[2]},\ \ldots,\\ &s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-2,\pi[n-2]},\ r \odot s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-2,\pi[n-2]},\\ &s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-1,\pi[n-1]},\ r \odot s \odot a_{1,\pi[1]} \odot \cdots \odot a_{n-1,\pi[n-1]}), \end{aligned}$$

with $\epsilon_i = e_i \odot a_{i,1} \oplus (g_0 \ominus e_i) \odot a_{i,0}$ and $\eta_i = f_i \odot (a_{i,1} \ominus a_{i,0})$.

Fig. 3. Matrices for the GLSPHF for $L'(s, H)$ (second case)

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 and no TCheck. Let us first consider the case with only one query **Hash** and no query **TCheck**. For that purpose, let us first prove the following theorem, which is an extension of Theorem 2 from [Sch01]. We point out that our result is slightly stronger than [MMN06, Theorems 2 and 3], because the bound is tighter and the adversary only needs to recover $g^{\hat{\pi}}$ and not $\hat{\pi}$.

Theorem 9. *Let $\mathbb{G}$ be a cyclic group of order p, and g a generator of $\mathbb{G}$. 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 β. Then, if $p - 2^n > 2^{2n}\sqrt{p}$, for any adversary $\mathcal{A}$ making at most t group operations[8]:*

$$\Pr\left[\mathcal{H} \xleftarrow{\$} \mathcal{D}_{\mathcal{H}}\ ;\ s \xleftarrow{\$} \mathbb{G}\ ;\ \tilde{\pi} \xleftarrow{\$} \mathcal{D}\ ;\ \hat{\pi} \leftarrow \mathcal{H}(\pi)\ ;\ H \leftarrow s^{\hat{\pi}}\ ;\ \mathcal{A}^{\mathcal{H}}(s, H) = g^{\hat{\pi}}\right] \leq \frac{4t-3}{2^\beta} + \frac{2^{2n}}{p} + \frac{3}{p} + \frac{t(t-1)}{2p}.$$

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).

The last term $\frac{t(t-1)}{2p}$ just comes from the fact the adversary ouputs $g^{\hat{\pi}}$ instead of $\hat{\pi}$, and is always negligible.

Remark 10. 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 2.3, 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.

[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.

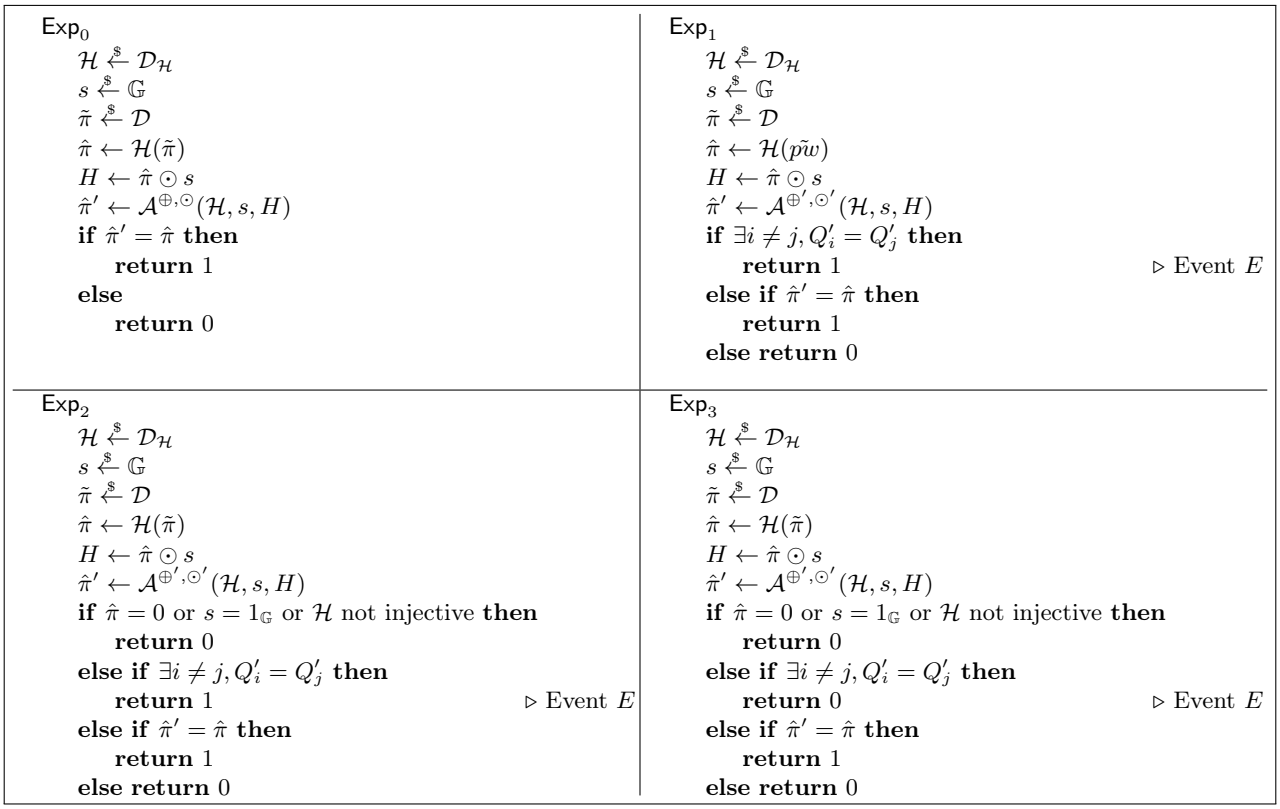

Fig. 4. Experiments for Theorem 9

Proof. Let us first suppose the adversary has to return $\hat{\pi}$ and not $g^{\hat{\pi}}$, and that g is not known to the adversary.

First of all, let us use an additive notation (as for graded rings): for any $x \in \mathbb{Z}_p$ and $u, v \in \mathbb{G}$: $x \odot u = u^x$, $u \oplus v = u \cdot v$. 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 Exp_0 the experiment implicitely considered in the theorem and depicted in Figure 4. 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:

$$Q_i \coloneqq x_i \odot s \ominus y_i \odot H,$$

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 $\hat{\pi} \odot s = s^{\hat{\pi}}$. Let us then consider the experiment Exp_1 which is similar to 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 4), 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 $\hat{\pi} \odot s$. 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.

If the event E does not happens Exp_1 and Exp_0 are identical, and otherwise, the adversary always wins in Exp_1. Clearly:

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

[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).

Let us now consider the experiment Exp_2 which is similar to Exp_1, except we abort (the adversary loses) when $\hat{\pi} = 0$ 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 $\Pr[\,\hat{\pi} = 0 \text{ or } s = 1_{\mathbb{G}}\,] \leq 2/p$ and the probability that $\mathcal{H}$ is not injective is:

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

Let us now consider the experiment Exp_3 which is similar to 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 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 $s^{\hat{\pi}}$. This happens if and only if $(x_i - x_j) = (y_i - y_j)\hat{\pi}$. If $(x_i - x_j) = (y_i - y_j)\hat{\pi}$, $y_i \neq y_j$ 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, Exp_0 and Exp_1 are distinct only if $\hat{\pi} \in S$. Since $\hat{\pi} \in \mathcal{H}(\{0,1\}^n)$ this is equivalent to $\hat{\pi} \in T$. 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]:

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

It remains to bound $|T|$. Let us first give a quick summary of the proof, before giving the details. First of all, there are 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:

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

[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 $|T|$.

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

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

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 Exp_2 and Exp_3), we get:

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

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:

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

This concludes the proof, when the adversary has to return $\hat{\pi}$ instead of $g^{\hat{\pi}}$.

We can easily adapt the proof when the adversary can return $g^{\hat{\pi}}$. The only difference is that polynomials Q_i also may contain a component in g:

$$Q_i = z_i \odot g \oplus x_i \odot s \ominus y_i \odot H,$$

and we define Q'_i to be the polynomials Q_i when H is replaced by $\hat{\pi}\mathsf{ts}\odot g$, and s is replaced by $\mathsf{ts}\odot g$, where ts is the discrete logarithm of s. Everything works as before except the bound of $\mathsf{Succ}_{\mathsf{Exp}_2} - \mathsf{Succ}_{\mathsf{Exp}_3}$, where we need to bound the probability that $Q'_i = Q'_j$ (while $Q_i \neq Q_j$). If $Q'_i = Q'_j$, we have: $(z_i - z_j) + (y_i - y_j)\mathsf{ts} = (x_i - x_j)\mathsf{ts}\hat{\pi}$. Thus, we have two cases:

- either $x_i = x_j$, they we get $(z_i - z_j) + (y_i - y_j)\mathsf{ts} = 0$, which happens with probability $1/(p-1)$, since ts is randomly chosen in $\mathbb{Z}_p^*$, and $y_i - y_j \neq 0$, otherwise $z_i = z_j$ and $Q_i = Q_j$.
- or $x_i \neq x_j$, and our previous bound works.

So we just get to add an additive factor $t(t-1)/(2p)$ to the previous bounds (since there are at most $t(t-1)/2$ pairs (Q_i, Q_j), with $Q_i \neq Q_j$). □

Case with One Hash Queries. Let us just quickly deal with the case where the adversary has access to a **TCheck** oracle. This oracle actually just allows the adversary to provide multiple possible answers P, in a slight variant of the experiments $\mathsf{Exp}_0, \ldots, \mathsf{Exp}_3$ where only $P = g^{\hat{\pi}}$ has to be provided instead of $\hat{\pi}$. So Equation (3) becomes:

$$\mathsf{Succ}_{\mathsf{Exp}_3} \le t2^{-\beta},$$

since there are at most t possible values P the adversary can generate. So we just loses a term $(t-1)2^{-\beta}$ in the bound.

Case with Multiple Hash Queries. We suppose for the sake of simplicity that the adversary returns $\hat{\pi}$ instead of $g^{\hat{\pi}}$, and that there is no **TCheck** queries. Both these minor points can be handled as previously.

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 $\hat{\pi}_k = \mathcal{H}(\tilde{\pi}_k)$ and $H_k = s_k^{\hat{\pi}_k}$.

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 $y_{k,j}$ ($1 \le j \le t$ and $1 \le k \le q$).

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 $\hat{\pi}_k \odot s_k$ (for all k); and in Exp_2 and Exp_3, we abort when one of the $\hat{\pi}_k$ 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 $x_{k,j} - x_{k,j'} = (y_{k,j} - y_{k,j'})\hat{\pi}_k$ for all k. Let us now consider the graph whose nodes are the polynomials Q_j. There is an edge, labeled by a set $S_{j,j'} = \{(k_{j,j',1}, \hat{\pi}'_{j,j',1}), \ldots, (k_{j,j',q'}, \hat{\pi}'_{j,j',q'})\}$, with $1 \leq k_{j,j',1} < \cdots < k_{j,j',q'} \leq q$ and $\hat{\pi}'_j \in \mathbb{Z}_p$, between Q_j and Q'_j, with $j \neq j'$, if:

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

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

Let us first handle the case when some non-singleton label $S_{j,j'}$ is compatible with $(\hat{\pi}_k)$. There are at most $t(t+1)/2$ distinct $S_{j,j'}$, and the probability that $(\hat{\pi}_k)$ 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:

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

the event E happens either for some non-singleton label $S_{j,j'}$ (case already handled) or when $\hat{\pi}_i \in S_i$ 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 $\hat{\pi}_j \in \mathcal{Q}$). 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 $\hat{\pi}_i \in S_i$, then $\hat{\pi}_i \in T'_i$, and $\hat{\pi}_i$ are distributed according to a distribution of min-entropy β (because $\mathcal{H}$ is supposed to be injective here). Therefore, the probability that $\hat{\pi}_i \in T'_i$ 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 $S_{j,j'} = \{(i, \hat{\pi}')\}$ 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 Multilinear Maps

In this section, we prove our password hashing scheme based on multilinear maps (Section 5.2) is secure in the generic graded ring model.

Statistical properties.

- *Correctness* and *salt indistinguishability* are straightforward.
- *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 generic graded ring 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 multilinear 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 β. The basic idea is to remark that the adversary queries to the oracles can be seen as

polynomials:

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

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 $a_{1,\tilde{\pi}_k[1]} \odot \cdots \odot a_{n,\tilde{\pi}_k[n]} \odot s_k$.

Case $q = 1$. 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 π 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 π. 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 $Q_{j_1},\dots,Q_{j_\ell},Q_{j_{\ell+1}} = Q_{j_1}$ with edges with distinct labels $\pi_1,\dots,\pi_\ell$ 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 $\pi_1,\dots,\pi_\ell$ are distinct and so all the monomials $a_{1,\pi_k[1]} \odot \cdots \odot a_{n,\pi_k[n]} \odot s_1$ (for $k = 1,\dots,\ell$) 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 $t/2^\beta$.

Case $q \geq 2$. Let us now analyse the case $q \geq 2$ and show that the adversary's advantage is approximatively at most $2t/2^\beta$, in all cases.

Let us consider the graph whose nodes are the polynomials Q_j. There is an edge, labeled by a set $S_{j,j'} = \{(k_{j,j',1},\pi_{j,j',1}),\dots,(k_{j,j',q'},\pi_{j,j',q'})\}$, 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 $k = 1,\dots,q'$); 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 $i = 1,\dots,q'$, 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 $Q_{j_1},\dots,Q_{j_\ell},Q_{j_{\ell+1}} = Q_{j_1}$ with edges of labels $\{(k_1,\pi_1)\},\dots,\{(k_\ell,\pi_\ell)\}$ 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 be the number 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 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. We also consider forward-secrecy and related-passwords as modeled in Section 3.2.

The proof consists in proving that the real attack game $\mathbf{G}_{\mathtt{real}} = \mathbf{G}_0$ is indistinguishable from an ideal one $\mathbf{G}_{\mathtt{ideal}}$, where the actual passwords are never used by the simulator before any corruption, or the end of the game.

Let us consider a polynomial time adversary $\mathcal{A}$, that wins (success) when its guess b' is equal to the challenge bit b. The proof is done by a series of games $\mathbf{G}_i$, and $\mathsf{Adv}_i(\mathcal{A}, \mathfrak{K}) = 2 \cdot \mathsf{Succ}_i(\mathcal{A}, \mathfrak{K}) - 1$ 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 recall that the adversary can only statistically corrupt users, i.e., it can corrupt a password or password hash when no instance of the associated players is involved in an execution of the protocol.

We separate `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 the salts are generated: we now use PTSalt instead of PSalt, so that we know a trapdoor ts for any salt s. Under salt indistinguishability, we have $|\mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A})| = 0$.

Game $\mathbf{G}_2$: Next, we modify the way `Execute` queries between two compatible users are answered. Since the hashing keys are known, we compute the common session key as

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

which does not change anything thanks to the correctness of the SPHFs. 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 `IND-CPA` property of the encryption schemes, for each `Execute` query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})| \leq \mathrm{negl}$.

Game $\mathbf{G}_3$: We modify again the way `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: $|\mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A})| \leq \mathrm{negl}$.

Game $\mathbf{G}_4$: We now modify the way `Execute` between two incompatible users are answered: we replace both session keys

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

(for the client and the server) by two independent truly random values. Thanks to the smoothness of the SPHFs, $\mathsf{Hash}(\mathsf{hk}_C, L_{H_S}, (\ell_S, c_S))$ and $\mathsf{Hash}(\mathsf{hk}_S, L'_{(s_S,H_S)}, c_C)$ are (close to) completely independent random values. And we have $|\mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A})| \leq \text{negl}$.

Game $\mathbf{G}_5$: We modify again the way `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 `IND-CPA` property of the encryption schemes, for each `Execute` query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A})| \leq \text{negl}$.

Game $\mathbf{G}_6$: We modify now the way we answer the $\mathtt{Send}_1$ queries, by using a decryption oracle, or alternatively knowing the decryption key. More precisely, when a message (hp_C, c_C) is sent, in the name of some client instance C^i and to some server instance S^j, if the password hash H_S of S is corrupted, we answer honestly (using s_S and H_S) and compute the session key honestly. Otherwise, three cases can appear:

- the message 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 $\mathsf{PTCheck}(\mathsf{param}, s_S, \mathsf{ts}_S, P, H_S) = 1$ with s_S the salt, ts_S its trapdoor, and H_S the hash value of the server S, then we choose the session key $K = \perp$; if later the adversary ask the session key via a `Test`-query, we stop the simulation and let it win;
  - 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 hk_C associated with hp_C, and we compute the session key K using the hashing key hk_C instead of 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 thanks to the correctness of PTCheck which ensures that if it outputs 0, the word is outside the language of the SPHF) 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: $\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A}) + \text{negl}$.

Game $\mathbf{G}_7$: 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, when the password hash of the server S is not corrupted. So, in this case, we can simply encrypt the dummy password 0 instead of the correct password π_S in all ciphertexts c_S, generated as responses to $\mathtt{Send}_1$ queries. This is indistinguishable under the `IND-CPA` property of the encryption scheme Enc_2: $|\mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A})| \leq \text{negl}$.

Game $\mathbf{G}_8$: We now modify the way the $\mathtt{Send}_2$ queries are answered, similarly to what we have done for the $\mathtt{Send}_2$ queries in $\mathbf{G}_8$. More precisely, when a message (s, hp_S, c_S) is sent, in the name of some server instance S^i and to some client instance C^j, if the password π_C of C has been corrupted, we answer honestly (using π_C) and compute the session key honestly. Otherwise, four cases can appear:

- the message is not a message generated by S (via a $\mathtt{Send}_1$ query) after receiving the first flow (hp_C, c_C) sent by C^j (via a $\mathtt{Send}_0$ query), then we first decrypt the ciphertext to get the hash value H used be the adversary:
  1. if $H = \mathsf{PHash}(\mathsf{param}, s, \pi_C)$ (i.e., if the password π_C is compatible with (s, H)) then we choose the session key $K = \perp$; if later the adversary ask the session key via a `Test`-query, we stop the simulation and let it win;
  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: $\mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) + \text{negl}$.

Game $\mathbf{G}_9$: In $\mathbf{G}_8$, 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, when the password of the client C is not corrupted. So, we can simply encrypt the dummy password 0 instead of the correct password π_C in all ciphertexts c_C, generated

as responses to $\texttt{Send}_0$ queries. This is indistinguishable under the IND-CCA property of the encryption scheme Enc_1: $|\mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A})| \leq \text{negl}$.

Game $\mathbf{G}_{10}$: We now modify the way we answer the $\texttt{Send}_1$ queries for replayed messages. More precisely, when a message (hp_C, c_C) is replayed by the adversary, we choose the session key K at random (except when the password π_C of C was corrupted, in which case the flow is honestly generated and the session key too). 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 hk_S looks completely random (given only hp_S): $|\mathsf{Adv}_{\mathbf{G}_{10}}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A})| \leq \text{negl}$.
We remark that, in this game, all session keys returned by the Test queries are completely independent and random (except for partnered compatible users, for which they are equal).

In this last game, the simulator does not use the passwords, and so no information leaks except in case of corruption. We also remark that the adversary wins only:

1. if it sends a flow to a server S (with non corrupted password hash), from which the simulator extracts a valid pre-hash P, such that $\mathsf{PTCheck}(\mathsf{param}, s_S, \mathsf{ts}_S, P, H_S) = 1$; and if the adversary makes a Test-query for that session (see $\mathbf{G}_6$)
2. or if it sends a flow to a client C, from which the simulator extracts a valid hash value H, such that $H = \mathsf{PHash}(\mathsf{param}, s, \pi_C)$, where s may be chosen by the adversary (see $\mathbf{G}_8$); and if the adversary makes a Test-query for that session (see $\mathbf{G}_6$)

If passwords are chosen independently at random from a distibution $\mathcal{D}$ of min-entropy β (i.e., not in the related-password model), and if no corruption occurs, then, for each flow, the first event happens with probability at most $2^{-\beta} + \text{negl}$ thanks to the second pre-image property (which ensures that $P = \mathsf{PPreHash}(\mathsf{param}, \pi_S)$ with π_S the password used to generate the hash value H_S of the server S) and the pre-hash entropy preservation (which ensures that finding this P knowing nothing about π_S nor H_S cannot be done with probability more than $2^{-\beta} + \text{negl}$). And the second event also happens with probaiblity at most $2^{-\beta} + \text{negl}$ for each flow, thanks to the entropy preservation.

So finally, with no corruption and independently chosen passwords, the probability of the adversary to win this last game (and also the original game) is at most:

$$q_s \times 2^{-\beta} + \text{negl}(\mathfrak{K})$$

with q_s the number of active sessions (sessions for which the adversary has sent a non-honestly-generated flow).

Now, if we allow client and server corruptions and if the the adversary never does a Test-query for a client with a corrupted password hash, everything is as above. Otherwise, in addition to the cases above, the adversary wins if we can extract from its flow to a server S with corrupted password hash H_S, a valid pre-hash P for H_S. But then we can construct an adversary $\mathcal{B}$ against the tight one-wayness of the password hashing such that

$$q_s \times 2^{-\beta} + \mathsf{Adv}_{\texttt{one-way}}(\mathcal{B}, \mathfrak{K}) + \text{negl}(\mathfrak{K}).$$

In case of related passwords, everything is similar as above, except the min-entropy of each password is computed differently (see Section 3.2).