2015 IEEE Symposium on Security and Privacy

Security of the J-PAKE Password-Authenticated Key Exchange Protocol

Michel Abdalla
CNRS, ENS, INRIA, and PSL
45 Rue d'Ulm, 75005 Paris, France
Email: michel.abdalla@ens.fr

Fabrice Benhamouda
ENS, CNRS, INRIA, and PSL
45 Rue d'Ulm, 75005 Paris, France
Email: fabrice.ben.hamouda@ens.fr

Philip MacKenzie
Google, Inc.
Mountain View, CA 94043
Email: philmac@google.com

***Abstract*—J-PAKE is an efficient password-authenticated key exchange protocol that is included in the OpenSSL library and is currently being used in practice. We present the first proof of security for this protocol in a well-known and accepted model for authenticated key-exchange, that incorporates online and offline password guessing, concurrent sessions, forward secrecy, server compromise, and loss of session keys. This proof relies on the Decision Square Diffie-Hellman assumption, as well as a strong security assumption for the non-interactive zero-knowledge (NIZK) proofs in the protocol (specifically, simulation-sound extractability). We show that the Schnorr proof-of-knowledge protocol, which was recommended for the J-PAKE protocol, satisfies this strong security assumption in a model with algebraic adversaries and random oracles, and extend the full J-PAKE proof of security to this model. Finally, we show that by modifying the recommended labels in the Schnorr protocol used in J-PAKE, we can achieve a security proof for J-PAKE with a tighter security reduction.**

I. Introduction

In a password-authenticated key exchange (PAKE) protocol, two parties who share only a password (i.e., a short secret) communicate with each other to compute a cryptographically strong shared secret key, using the password for mutual authentication. The protocol should not allow an attacker to obtain any information about the password through simple eavesdropping, and only allow the attacker to gain information about one password per protocol session in an active attack. Basically, this implies that the attacker is not able to obtain data with which to perform an *offline dictionary attack*, in which the attacker would run through a dictionary of possible passwords offline, checking each one for consistency with the data. A very good introduction and discussion of this problem may be found in Jablon [29] or Wu [47]. The seminal work in the field was the development of Encrypted Key Exchange (EKE) by Bellovin and Merritt [7], [8], and there has been a great deal of work since then (for references see, e.g., [28]).

The J-PAKE protocol [24] is a PAKE protocol that has started seeing wide usage. It is included as an optional protocol in the OpenSSL library [39] (enabled using a config parameter during install, see directory crypto/jpake), and has been used in various products, such as Firefox Sync [16] and Nest products [38] (as part of the Thread protocol [46]). Its popularity is likely due not only to its easy description, straightforward implementation, and practical efficiency, but also because it seems to be based on a different paradigm than previous practical PAKE protocols. Those protocols basically used the password to obfuscate the inputs to a key exchange (e.g., the g^x and g^y values in a Diffie-Hellman key exchange), whereas the J-PAKE protocol uses ephemeral values like a standard Diffie-Hellman key exchange, but then combines them with a password in an extra round, such that use of the correct password makes certain randomization factors vanish. The J-PAKE designers call this the "juggling" technique and attribute the first use of the idea to Hao and Zielinski [25]. Due to its novelty, the designers of J-PAKE claim that it might be useful in avoiding patent issues around other PAKE protocols.

The original J-PAKE paper claimed to give a proof of security, but, as pointed out by Katz [31], the proof was not in one of the well-known accepted models for authenticated key exchange (e.g., the model from Bellare, Pointcheval, and Rogaway [5]), and simply proved some ad-hoc properties in an isolated setting, using implicit assumptions on the adversarial model. Given its growing popularity, it is important to have a better understanding of the security of this protocol, using rigorous and explicit definitions and models. This is especially true for PAKE protocols, since there are many subtleties to their security, and many previous PAKE protocols, or early versions of PAKE protocols (that did not have rigorous security proofs) have been shown to be insecure [36], [41].

In this paper we present a proof of security for the J-PAKE protocol in the well-known authenticated key exchange model of Bellare, Pointcheval, and Rogaway [5], under the Decision Square Diffie-Hellman (DSDH) assumption, along with other assumptions described below. The DSDH assumption is similar to and at least as strong as the Decision Diffie-Hellman assumption, but it is not known whether it is strictly stronger. We note that we could reduce this assumption to DDH and Computational Square Diffie-Hellman (CSDH)[1] by using the random-oracle model.[2]

One interesting technique used in the J-PAKE protocol that has not been used in previous PAKE proofs is the zero-

[1]Since there is a reduction from DDH to CSDH, we could say this is only based on DDH. However, there is a quadratic loss in concrete security, so we prefer to keep the assumptions separate.

[2]Bellare and Rogaway [6] introduced the random-oracle model in which hash functions are modeled as random oracles, and argue that proofs in such an ideal model provide evidence that when the ideal constructs are instantiated properly (with strong cryptographic implementations), then the protocol remains secure in practice.

DOI 10.1109/SP.2015.41

knowledge (ZK) proof of knowledge. Generally it is difficult to argue about the security of ZK proofs of knowledge in a concurrent protocol model. This is because for most known ZK proofs of knowledge, and even non-interactive ZK (NIZK) proofs of knowledge in the random-oracle model, rewinding arguments have been used to prove the extraction property, which is problematic in a concurrent setting since it can cause an exponential expansion in simulation cost during reduction arguments. We initially avoid this issue and assume the use of NIZK proofs of knowledge that are simulation-sound extractable [22], with non-rewinding extractors. We call these SE-NIZK proofs. One could say that this proves the security of J-PAKE in a rigorous model that captures the standard intuition behind NIZK proofs of knowledge, and more specifically, proves security under the DSDH assumption and the assumptions necessary to prove the internal NIZK proof of knowledge is simulation-sound extractable.

However, the NIZK proof of knowledge recommended by the designers of J-PAKE (and used in the current implementations) is the Schnorr protocol [43], which seems to require rewinding arguments to prove the extraction property, at least in the standard computation model. Therefore, to provide a rigorous proof of security of J-PAKE using the Schnorr protocol, we turn to the algebraic model [40] (with respect to a group G), in which an adversary is limited to perform only group operations on group elements in G. It is similar to the generic group model of Shoup [44], in which all group operations are performed using an oracle, but is weaker as, in particular, it makes no assumption on the representation of group elements and does not imply by itself that, e.g., the discrete logarithm is hard. We show that in the algebraic model, the Schnorr protocol can be seen as an SE-NIZK proof, in any proof by reduction, with some restrictions (on the group elements used by the proof) that our J-PAKE proof does in fact satisfy. This proof relies on the Discrete Log (DL) assumption in the random-oracle model. Putting this all together, we have proven the security of J-PAKE using Schnorr in the algebraic model and random-oracle model, under the DSDH assumption. It is worth emphasizing that this is a proof of security that matches the underlying implementation in OpenSSL, and this is important in that it allows applications to use J-PAKE in a way that exactly matches the security proof.[3]

Returning to the standard computation model, Groth, Ostrovsky, and Sahai [23] and Groth [22] show how to achieve SE-NIZK proofs in the common reference string (CRS model). Garay, MacKenzie, and Yang [17] and MacKenzie and Yang [37] show how to achieve non-malleable ZK proofs (which are like SE-NIZK proofs but allowed to be interactive) which trivially imply SE-NIZK proofs in the CRS and random-oracle model, and require only a constant number of exponentiations (but over multiple groups with larger non-prime moduli). Any of these could replace the Schnorr proof of knowledge in the J-PAKE protocol, though none of them would be nearly as practical.

As a final result, we show that by slightly modifying the labels used in the Schnorr proofs in the J-PAKE protocol, one can obtain a simpler security proof, with tighter security reductions from known cryptographic assumptions. We recommend using these modified labels in future implementations of the J-PAKE protocol, if they don't require backwards compatibility.

Other PAKE protocols. Many previous practical PAKE protocols have been proven secure in either the random-oracle model or ideal-cipher model, e.g., [3], [5], [7], [10], [29], [35], [36]. As shown in [15], [27], the ideal-cipher model is equivalent to the random-oracle model, when the inputs and outputs are binary strings. In practice, however, ideal ciphers for group elements, as required in [5], [7], are difficult to construct and can have an impact on the efficiency of the schemes. In addition, a few PAKE protocols have been proven secure without ideal assumptions. For instance, the practical protocol of Katz, Ostrovsky, and Yung [32] only relies on a reasonably short common reference string that is produced before the protocol begins. This protocol has been generalized and improved in several follow-up works, such as [1], [12], [18], [21], [30], [33]. For these protocols, the common reference string could be simulated using a random oracle. The protocol of Goldreich and Lindell [19] does not rely on a common reference string either, but is only proven secure when protocols sessions are *not* run concurrently, and does not seem practical. More recently, Goyal, Jain, and Ostrovsky [20] improved the work of Goldreich and Lindell by providing a protocol that is proven secure even when protocols sessions are run concurrently. Their protocol also does not seem practical. A detailed comparison of practical Diffie-Hellman-based PAKE protocols can be found in Table I.

II. Definitions

Let κ be the cryptographic security parameter. Let G denote a finite (cyclic) group of order p, where $|p| \geq 2\kappa$. Let g be a generator of G. We will assume the Decision Square Diffie-Hellman (DSDH) assumption holds over G (see Section VI). Let t_{exp} be the time required to perform an exponentiation in G.

If $\mathsf{Adv}(\mathcal{A})$ denotes the advantage of some adversary in some experiment, we write $\mathsf{Adv}(t) = \max_{\mathcal{A}} \{\mathsf{Adv}(\mathcal{A})\}$, where the maximum is taken over all adversaries of time complexity at most t.

A. Random Oracle

In some cases, cryptographic hash functions will be modeled as random oracles. Thus whenever a party computes a function $H(x)$, the party is actually sending a query x to the random oracle designated for H, and that oracle returns $H(x)$, where H is a truly random function with domain $\{0,1\}^*$ and range $\{0,1\}^\kappa$ or $\mathbb{Z}_p$ depending on the case.

[3] Applications do have some flexibility in how certain labels are chosen within the protocol, which may affect the security. We discuss this after the proof in Section VI.

TABLE I
COMPARISON OF PRACTICAL DIFFIE-HELLMAN-BASED PAKE PROTOCOLS PROVEN SECURE IN THE BPR MODEL [5]

		Assumptions[a]					Complexity	
	Rounds / Flows	CRS	ROM	ICM	AAM		Communication[b]	Time[c]
J-PAKE with Schnorr [24]	2 / 4 or 3 / 3		✗		✗	DSDH or (CSDH + DDH)	$12 \times G + 6 \times \mathbb{Z}_p$	28 exp (12 exp + 8 mexp)
EKE [5], [7]	1 / 2			✗		CDH	$2 \times G$	4 exp + 2 memb + 2 enc
SPEKE [29], [35]	1 / 2		✗			DIDH[d]	$2 \times G$	4 exp + 2 memb
PPK [10]	2 / 2		✗			DDH	$2 \times G$	6 exp + 2 memb
SPAKE2 [3]	1 / 2		✗			CDH	$2 \times G$	4 exp + 2 memb
GK-SPOKE [1], [21], [30]	2 / 2	✗				DDH + PRG[e]	$6 \times G$	17 exp (4 exp + 7 mexp) + 6 memb
GL-SPOKE [1], [18], [32]	2 / 2	✗				DDH	$7 \times G$	21 exp (4 exp + 7 mexp) + 7 memb
KV-SPOKE [1], [33]	1 / 2	✗				DDH	$10 \times G$	30 exp (2 exp + 12 mexp) + 10 memb

[a] *CRS*: common reference string, *ROM*: random-oracle model, *ICM*: ideal-cipher model, *AAM*: algebraic-adversary model;
[b] G: group elements, $\mathbb{Z}_p$: scalars;
[c] *exp*: number of exponentiations; *mexp*: number of multi-exponentiations; *memb*: verification of the membership of a group element to the cyclic group G. For elliptic curve with small co-factor, this only costs a small number of additions on the curve, but for subgroups of $\mathbb{Z}_q$ (q being a prime larger than p, the order of the group G), this costs an exponentiation (with exponent $p-1$); *enc*: encryption with the ideal cipher; multiplications, hash evaluations, and PRG evaluations are omitted;
[d] *DIDH*: decision inverted-additive Diffie-Hellman assumption [35] (see Fig. 2 and the Appendix);
[e] *PRG*: pseudo-random generator.

B. Simulation-Sound Extractable Non-Interactive Zero-Knowledge Proofs

We will assume the zero-knowledge proofs of knowledge in the J-PAKE protocol are simulation-sound extractable non-interactive zero-knowledge proofs (SE-NIZK) [22].

Informally, a SE-NIZK enables a prover to prove that some word or statement x is in a given NP-language $\mathscr{L}$, defined with some witness relation $\mathcal{R}$, i.e., $\mathscr{L} = \{x \mid \exists\omega,\ \mathcal{R}(x,\omega) = 1\}$, in a zero-knowledge and extractable way. If ω is such that $\mathcal{R}(x,\omega) = 1$, ω is said to be a witness for x. Given, a witness ω for x, the prover can generate a proof $\pi \xleftarrow{R} \mathsf{PK}(x,\omega,\ell)$ for some label ℓ (and maybe some implicit common reference string or CRS σ). This proof can be checked by anyone by running an algorithm $\mathsf{VK}(x,\pi,\ell)$. We insist that we introduce an optional CRS for the sake of completeness, but that for the instantiation of J-PAKE using Schnorr proofs, no CRS is used.

In addition, knowing some trapdoor τ, it is possible to simulate any proof for any word x (even outside $\mathscr{L}$) without knowing a witness ω. And, knowing some other trapdoor ξ, it is possible to extract from any valid proof π for any word x, a witness ω for x. Simulation-sound extractability ensures that the extraction works even if the adversary sees simulated proofs (except for the simulated tuples (x,π,ℓ) obviously).

Concretely, in this article, we only consider SE-NIZK proofs for the language of discrete logarithms: $x = (u,h) \in G^2$, $\omega = r \in \mathbb{Z}_p$, and

$$\mathcal{R}((u,h),r) = 1 \iff u = h^r.$$

In other words, our SE-NIZK proofs are proofs of knowledge of the discrete logarithm of u in base h.

Formal definitions can be found in Section VIII.

We should point out that we assume the extractor is straight-line: no rewinding is authorized. This rules out using directly Schnorr proofs, also known as Schnorr signatures [43], (which are the most efficient NIZK for the discrete logarithm language, in the random-oracle model) with a rewinding extractor based on the forking lemma [42]. Nevertheless, in Section IX, we show that Schnorr signatures can still be used in J-PAKE, if we assume adversaries are algebraic. This, however, requires a careful analysis and the introduction of a weaker form of simulation-sound extractability, called algebraic-simulation-sound extractability.

C. Computational Randomness Extractor

The last step of the original J-PAKE protocol consists in deriving a secret key from a group element. In the original paper, this is done by using a hash function implicitly modeled as a random oracle. In this paper, we prefer to formally define the requirements for this derivation function. More precisely we suppose it is a (computational) randomness extractor [34], for random group elements.

Note that if no randomness extractor is used in the J-PAKE protocol, the protocol would still be secure[4]. The only difference would be that the session key would be a random group element G instead of a random bitstring in $\{0,1\}^\kappa$.

A computational randomness extractor for (uniform) group elements is a function $\mathrm{rExt} : \{0,1\}^t \times G \to \{0,1\}^\kappa$, for some non-negative integer t, such that, when $s \xleftarrow{R} \{0,1\}^t$ and $u \xleftarrow{R} G$ (independent of s), then $(s, \mathrm{rExt}(s,u))$ is computationally indistinguishable from a uniform bit string in $\{0,1\}^t \times \{0,1\}^\kappa$. Formally, given a poly-time adversary $\mathcal{A}$, we consider the advantage $\mathsf{Adv}^{\text{comp-ext}}_{\mathrm{rExt}}(\mathcal{A})$ defined as:

$$\Pr\left[s \xleftarrow{R} \{0,1\}^t; u \xleftarrow{R} G : \mathcal{A}(s, \mathrm{rExt}(s,u)) = 1\right] - \Pr\left[s \xleftarrow{R} \{0,1\}^t; k \xleftarrow{R} \{0,1\}^\kappa : \mathcal{A}(s,k) = 1\right].$$

As shown in [34], a hash function H modeled as a random oracle gives a randomness extractor without seed ($s = \perp$, $t = 0$, $\mathrm{rExt}(\perp, u) = H(u) \in \{0,1\}^\kappa$) with $\mathsf{Adv}^{\text{comp-ext}}_{\mathrm{rExt}}(\mathcal{A}) \leq n_{\mathrm{ro}}/p$, with n_{ro} the number of queries to the random oracle.

[4] The proof would be the one in Section VI, without the last protocol P_8, and so would actually be slightly simpler.

Considering a hash function to be a randomness extractor is weaker than modeling it as a random oracle. However, if such an assumption is not deemed acceptable, there exist various alternative solutions: some with seeds such as the left over hash lemma [26], and some without seeds, but specific to some groups [13]. For further references on randomness extractors, see [14].

III. Model

For our proofs of security we use a real-or-random variant of the model of [5] with weak adaptive corruptions (corruption queries do not reveal the internal state of the principals, but reveal the password of the principal and can be made at any point during the protocol) and forward secrecy In [2], it is shown that this real-or-random variant is stronger than the original find-then-guess model in [5]. The only difference with [5] is that we allow multiple Test queries.

Protocol participants and long-lived keys. Participants in the protocol are either clients and servers. Each client A holds a password pw_A chosen uniformly (and independently) at random from a dictionary of size N. Each server B holds a vector of the passwords of all clients, and when running the protocol with some client A, uses the password pw_A of A. Users are modeled as probabilistic poly-time algorithms that respond to queries. For any user U, we will let U denote both the user, and the identifier for the user (e.g., to be used as input to a function).

Execution of the protocol. A protocol P is an algorithm that determines how principals behave in response to inputs from their environment. In the real world, each principal is able to execute P multiple times with different partners, and we model this by allowing unlimited number of *instances* of each principal. Instance i of principal U is denoted Π_i^U.

To describe the security of the protocol, we assume there is an adversary $\mathcal{A}$ that has complete control over the environment (mainly, the network), and thus provides the inputs to instances of principals. Formally, at the beginning of the protocol, a random bit b is chosen. The adversary is a probabilistic algorithm with a distinguished query tape. Queries written to this tape are responded to by principals according to P; the allowed queries are formally defined in [5] and summarized here:

- $\mathsf{Send}\ (U,\ i,\ M)$**:** causes message M to be sent to instance Π_i^U. The instance computes what the protocol says to, state is updated, and the output of the computation is given to $\mathcal{A}$. If this query causes Π_i^U to accept or terminate, this will also be shown to $\mathcal{A}$. To initiate a session between users A and B the adversary should send a message containing B to an unused instance of A, or a message containing A to an unused instance of B.
- $\mathsf{Execute}\ (A,\ i,\ B,\ j)$**:** causes P to be executed to completion between Π_i^A and Π_j^B, and outputs the transcript of the execution. This query captures the intuition of a passive adversary who simply eavesdrops on the execution of P. It could be simulated with Send queries, but having separate Execute queries enable to state stronger security results.
- $\mathsf{Reveal}\ (U,\ i)$**:** causes the output of the session key held by Π_i^U.
- $\mathsf{Test}\ (U,\ i)$**:** causes the output of the session key sk_U^i, if $b = 1$; otherwise, a string is drawn uniformly from the space of session keys and output.
- $\mathsf{Corrupt}\ (U)$**:** causes the client U to output its password.

Partnering. A client or server instance that accepts holds a partner-id pid, session-id sid (which is the transcript of the whole protocol), and a session key sk. Then instances Π_i^A and Π_j^B are said to be *partnered* if both accept, they hold (pid, sid, sk) and (pid', sid', sk'), respectively, with $pid = B$, $pid' = A$, $sid = sid'$, and $sk = sk'$, and no other instance accepts with session-id equal to sid.

Freshness. An instance Π_i^U is fresh unless either (1) a $\mathsf{Reveal}\ (U,\ i)$ query occurs, or (2) a $\mathsf{Reveal}\ (U',\ j)$ query occurs where $\Pi_{U'}^j$ is the partner of Π_U^i, or (3) a $\mathsf{Corrupt}\ (U')$ query occurs before Π_i^U defined its key sk_U^i, and a $\mathsf{Send}\ (U,\ i,\ M)$ query occurred, where U' is any participant.

Advantage of the adversary. We now formally define the authenticated key exchange (*ake*) advantage of the adversary against protocol P. Let $\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})$ be the event that $\mathcal{A}$ makes only Test queries directed to fresh instances Π_i^U that have terminated, and eventually outputs a bit b', where $b' = b$ for the bit b that was selected at the beginning of the protocol. The *ake* advantage of $\mathcal{A}$ attacking P is defined to be

$$\mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} 2\Pr\left[\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})\right] - 1.$$

The authenticated key exchange is considered secure if only online dictionary attacks are possible. Concretely, this means that, if passwords are uniformly and independently drawn from a dictionary of size N:

$$\mathsf{Adv}_P^{\mathrm{ake}}(t) \leq n_{\mathrm{se}}/N + \epsilon,$$

where n_{se} is the number of Send queries (to distinct instances Π_U^i), and ϵ is negligible in the security parameter:

The following fact is easily verified.

Fact III.1. $\Pr(\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})) = \Pr(\mathsf{Succ}_{P'}^{\mathrm{ake}}(\mathcal{A})) + \epsilon$ if and only if $\mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) = \mathsf{Adv}_{P'}^{\mathrm{ake}}(\mathcal{A}) + 2\epsilon$.

Both the model and our proofs can be extended (in a straightforward way) when the password distribution is not uniform but only has some min-entropy m. In this case, we want that $\mathsf{Adv}_P^{\mathrm{ake}}(t) \leq n_{\mathrm{se}}/2^m + \epsilon$.

IV. J-PAKE protocol

Here we present the J-PAKE protocol from [24].

A. Informal version

Informally, Alice and Bob, who share a password pw, do the following:

Round 1 Alice randomly generates g^{x_1}, g^{x_2}, Bob randomly generates g^{x_3}, g^{x_4}, Alice and Bob send these values to each

other, along with NIZK proofs for the exponents (denoted $\pi_1, \pi_2, \pi_3, \pi_4$ respectively).

Round 2 Alice sends $\alpha = (g^{x_1+x_3+x_4})^{x_2\mathsf{pw}}$ to Bob along with an NIZK proof π_α for the exponent $x_2\mathsf{pw}$. Bob sends $\beta = (g^{x_1+x_2+x_3})^{x_4\mathsf{pw}}$ to Alice along with an NIZK proof π_β for the exponent $x_4\mathsf{pw}$.

Session Key Generation, Alice computes $K = (\beta/g^{x_2x_4\mathsf{pw}})^{x_2}$, and Bob computes $K = (\alpha/g^{x_2x_4\mathsf{pw}})^{x_4}$. They both compute a session key $sk = \mathrm{rExt}(s, K)$, where s is a public random seed in $\{0,1\}^t$. This is a slight generalization of the original J-PAKE protocol, where there was no seed ($t = 0$) and $sk = \mathrm{rExt}(\perp, K) = H(K)$, with H being a hash function behaving as a randomness extractor.

B. Formal version

In Fig. 1, we present a fully specified version of the J-PAKE protocol.

While the protocol is written symmetrically for users A and B, clients and servers are disjoint, and thus only clients generate X_1, X_2, α, and only servers generate X_3, X_4, β.

We do not need to suppose any ordering between flows in each round. In particular, it is possible to merge the flow $(B, X_3, X_4, \pi_3, \pi_4)$ with the flow (β, π_β) to get a three-flow protocol, although it seems it was not done in concrete implementations [16], [39].

V. Security Assumptions

Here we state the assumptions we use for the remainder of the paper. We assume we have a cyclic group G of prime order p generated by element g. Relations between assumptions are summarized in Fig. 2 on Page 6.

Discrete Logarithm (DL). An algorithm for the discrete logarithm problem takes an element $X = g^x \in G$, and attempts to output its discrete logarithm x. Let $\mathcal{A}$ be an algorithm with input X. Let $\mathsf{Adv}_G^{\mathrm{DL}}(\mathcal{A})$ be

$$\Pr\left[x \xleftarrow{R} \mathbb{Z}_p;\, X \leftarrow g^x :\ \mathcal{A}(X) = x\right].$$

Computational Diffie-Hellman (CDH). For two values $X = g^x$ and $Y = g^y$, let $\mathsf{DH}(X, Y) = g^{xy}$ be the Diffie-Hellman value corresponding to X and Y. An algorithm for the Computational Diffie-Hellman takes two elements X and Y, and outputs the Diffie-Hellman value of X and Y. Let $\mathcal{A}$ be an algorithm with input (X, Y). Let $\mathsf{Adv}_G^{\mathrm{CDH}}(\mathcal{A})$ be

$$\Pr\left[(x,y) \xleftarrow{R} \mathbb{Z}_p^2;\, X \leftarrow g^x;\, Y \leftarrow g^y :\ \mathcal{A}(X,Y) = \mathsf{DH}(X,Y)\right].$$

Decision Diffie-Hellman (DDH). An algorithm for the Decision Diffie-Hellman takes three elements X, Y, and Z, and attempts to distinguish whether Z is the Diffie-Hellman value corresponding to X and Y, or is a random element of G. Let $\mathcal{A}$ be an algorithm with input (X, Y, Z). Let $\mathsf{Adv}_G^{\mathrm{DDH}}(\mathcal{A})$ be

$$\begin{aligned}\Pr\left[(x,y) \xleftarrow{R} \mathbb{Z}_p^2;\, X \leftarrow g^x;\, Y \leftarrow g^y;\, Z \leftarrow \mathsf{DH}(X,Y) :\right.\\ \left.\mathcal{A}(X,Y,Z) = 1\right]\\ - \Pr\left[(x,y,z) \xleftarrow{R} \mathbb{Z}_p^3;\, X \leftarrow g^x;\, Y \leftarrow g^y;\, Z \leftarrow g^z :\right.\\ \left.\mathcal{A}(X,Y,Z) = 1\right].\end{aligned}$$

See [9] for a discussion on hardness of DDH over various groups.

Note that DDH is random self-reducible. That is, from a single DDH tuple (X, Y, Z) we can generate any number of random independent DDH tuples with the same property (either DH or random) by choosing $a_1, b_1, b_2 \xleftarrow{R} \mathbb{Z}_p$, and generating $(X^{a_1}g^{b_1}, Yg^{b_2}, Z^{a_1}X^{a_1b_2}Y^{b_1}g^{b_1b_2})$.

Note also that DDH can be solved with one application of CDH, and CDH can be solved with one application of DL so $\mathsf{Adv}_G^{\mathrm{DDH}}(t) + 1/p \geq \mathsf{Adv}_G^{\mathrm{CDH}}(t) \geq \mathsf{Adv}_G^{\mathrm{DL}}(t)$.

Computational Square Diffie-Hellman (CSDH). For value $X = g^x$, let $\mathsf{SDH}(X) = g^{x^2}$ be the square Diffie-Hellman value corresponding to X. An algorithm for Computational Square Diffie-Hellman takes an element of G, and computes the square Diffie-Hellman value. Let $\mathcal{A}$ be an algorithm with input X. Let $\mathsf{Adv}_G^{\mathrm{CSDH}}(\mathcal{A})$ be

$$\Pr\left[x \xleftarrow{R} \mathbb{Z}_p;\, X \leftarrow g^x :\ \mathcal{A}(X) = \mathsf{SDH}(X)\right].$$

Note that CSDH is random self-reducible. That is, from a single instance of CSDH we can generate multiple random instances such that if we find the square of one, then we can find the square of the original instance. Given an instance X, generate a new instance $X' = Xg^r$ for $r \xleftarrow{R} \mathbb{Z}_p$. Then X' is a random instance, where $\mathsf{SDH}(X) = \mathsf{SDH}(X')X^{-2r}g^{-r^2}$.

Since the CDH problem can be solved using 2 SDH values of independently chosen elements [11], it is easy to show that for $t' = t + O(t_{\mathsf{exp}})$, $\mathsf{Adv}_G^{\mathrm{CDH}}(t) \geq (\mathsf{Adv}_G^{\mathrm{CSDH}}(t'))^2$, or equivalently, $\mathsf{Adv}_G^{\mathrm{CSDH}}(t) \leq (\mathsf{Adv}_G^{\mathrm{CDH}}(t'))^{1/2}$, where t_{exp} is the time for an exponentiation in G.

Decision Square Diffie-Hellman (DSDH). An algorithm for Decision Square Diffie-Hellman takes two elements X and Y, and attempts to distinguish wether Y is the square Diffie-Hellman value corresponding to X, or is a random element of G. Let $\mathcal{A}$ be an algorithm with input X. Let $\mathsf{Adv}_G^{\mathrm{DSDH}}(\mathcal{A})$ be

$$\begin{aligned}&\Pr\left[x \xleftarrow{R} \mathbb{Z}_p;\, X \leftarrow g^x;\, Y \leftarrow \mathsf{SDH}(X) :\ \mathcal{A}(X,Y) = 1\right]\\ &- \Pr\left[(x,y) \xleftarrow{R} \mathbb{Z}_p^2;\, X \leftarrow g^x;\, Y \leftarrow g^y :\ \mathcal{A}(X,Y) = 1\right].\end{aligned}$$

Bao *et al.* [4] show that $\mathsf{Adv}_G^{\mathrm{DDH}}(t) \leq \mathsf{Adv}_G^{\mathrm{DSDH}}(t')$, for $t' = t + O(t_{\mathsf{exp}})$.

Note that DSDH is random self-reducible. That is, from a single DSDH tuple (X, Y) we can generate any number of random independent DSDH tuples with the same property (either SDH or random) by choosing $a, b \xleftarrow{R} \mathbb{Z}_p$, and generating $(X^a g^b, Y^{a^2} X^{2ab} g^{b^2})$.

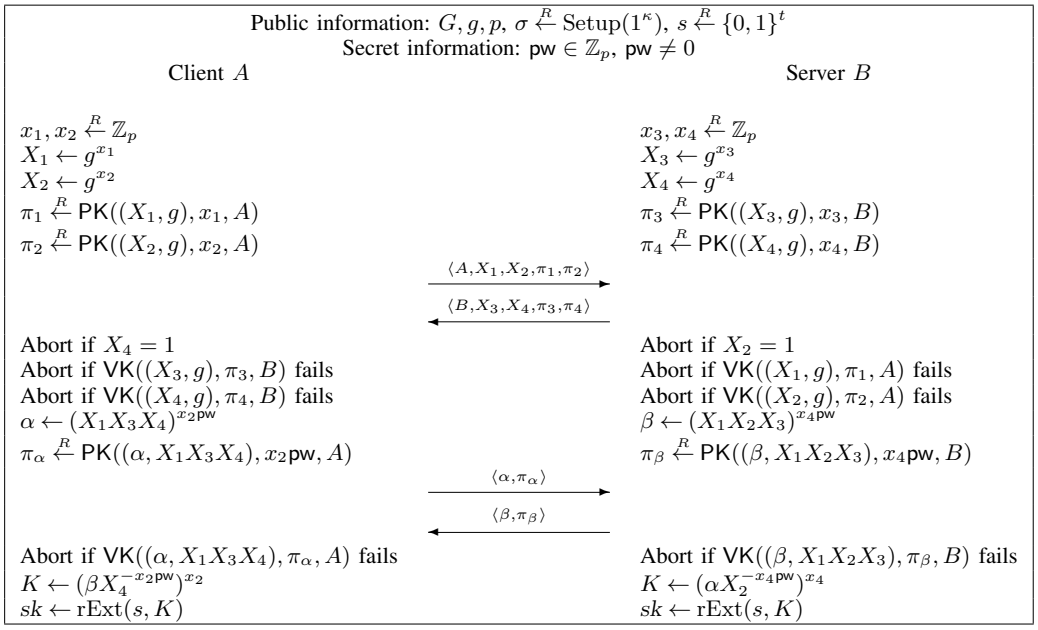

Fig. 1. Generalized version of J-PAKE in which a random extractor is used to derive sk from K. The original J-PAKE protocol is a particular case where $t = 0$, $s = \perp$, $\mathrm{rExt}(s, K) = H(K)$, and H is a hash function behaving as a randomness extractor.

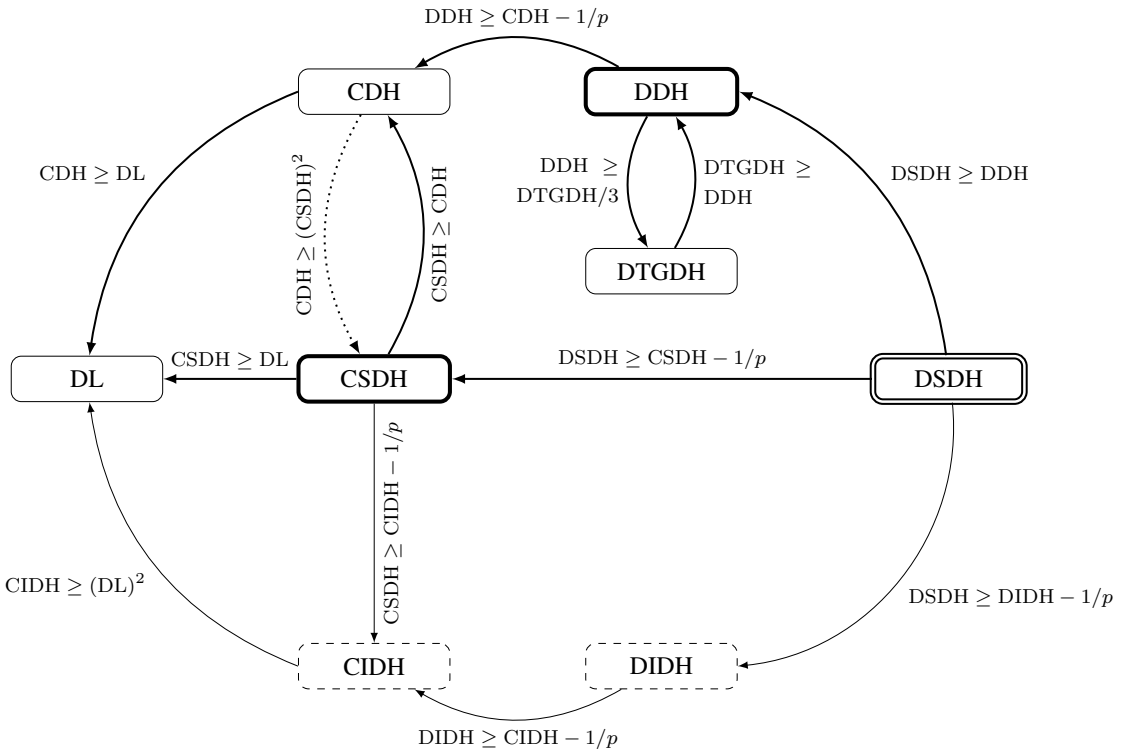

Fig. 2. Relations between assumptions. DL, CDH, … correspond to the advantage of an adversary for these problems ($\mathsf{Adv}^{\mathrm{DL}}_G(t)$, …). An arrow indicates an implication, e.g., the arrow from CDH to DL means that, if CDH is hard, so is DL. A dotted arrow means there is an important loss in the reduction (here a quadratic loss). Our proof for J-PAKE is either under DSDH (the strongest assumption), or under DDH and CSDH (slightly weaker assumptions) but in the random-oracle model. CIDH and DIDH are used to prove the security of the SPEKE protocol [29], [35] (see Table I and the Appendix).

Decision Triple Group Diffie-Hellman (DTGDH). For values $X = g^x$, $Y = g^y$, $Z = g^z$, let $\mathsf{TDH}(X, Y, Z) = g^{xyz}$ be the triple Diffie-Hellman value corresponding to X, Y, and Z. An algorithm for Decision Triple Group Diffie-Hellman takes a triple of elements of G, their pairwise Diffie-Hellman values, and an element W from G, and attempts to distinguish whether W is the Triple Diffie-Hellman value corresponding to X, Y, and Z, or is a random element of G. Let $\mathcal{A}$ be an algorithm with input $(X, Y, Z, D_{XY}, D_{XZ}, D_{YZ}, W)$. Let $\mathsf{Adv}_G^{\mathrm{DTGDH}}(\mathcal{A})$ be

$$\begin{aligned}\Pr\big[(x,y,z) \xleftarrow{R} \mathbb{Z}_p^3;\ X \leftarrow g^x;\ Y \leftarrow g^y;\ Z \leftarrow g^z;\\ D_{XY} \leftarrow \mathsf{DH}(X,Y);\ D_{XZ} \leftarrow \mathsf{DH}(X,Z);\\ D_{YZ} \leftarrow \mathsf{DH}(Y,Z);\ W \leftarrow \mathsf{TDH}(X,Y,Z):\\ \mathcal{A}(X,Y,Z,D_{XY},D_{XZ},D_{YZ},W) = 1\big]\\ -\Pr\big[(x,y,z,w) \xleftarrow{R} \mathbb{Z}_p^4;\ X \leftarrow g^x;\ Y \leftarrow g^y;\ Z \leftarrow g^z;\\ D_{XY} \leftarrow \mathsf{DH}(X,Y);\ D_{XZ} \leftarrow \mathsf{DH}(X,Z);\\ D_{YZ} \leftarrow \mathsf{DH}(Y,Z);\ W \xleftarrow{R} g^w:\\ \mathcal{A}(X,Y,Z,D_{XY},D_{XZ},D_{YZ},W) = 1\big].\end{aligned}$$

Note that DTGDH is random self-reducible. That is, from a single DTGDH tuple

$$(X, Y, Z, D_{XY}, D_{XZ}, D_{YZ}, W)$$

we can generate any number of random independent DTGDH tuples with the same property (either TDH or random) by choosing $a, b_1, b_2, b_3 \xleftarrow{R} \mathbb{Z}_p$, and generating

$$(X', Y', Z', D'_{XY}, D'_{XZ}, D'_{YZ}, W'),$$

where

$$\begin{aligned}X' &\leftarrow X^a g^{b_1}, & D'_{YZ} &\leftarrow D_{YZ} Y^{b_3} Z^{b_2} g^{b_2 b_3},\\ Y' &\leftarrow Y g^{b_2}, & D'_{XZ} &\leftarrow (D_{XZ})^a X^{ab_3} Z^{b_1} g^{b_1 b_3},\\ Z' &\leftarrow Z g^{b_3}, & D'_{XY} &\leftarrow (D_{XY})^a X^{ab_2} Y^{b_1} g^{b_1 b_2},\\ W' &\leftarrow W^a (D_{XY})^{ab_3} (D_{XZ})^{ab_2} (D_{YZ})^{b_1}\\ & & & X^{ab_2 b_3} Y^{b_1 b_3} Z^{b_1 b_2} g^{b_1 b_2 b_3}.\end{aligned}$$

DTGDH is also random partial self-reducible, in that from a single DTGDH tuple

$$(X, Y, Z, D_{XY}, D_{XZ}, D_{YZ}, W)$$

we can generate any number of random DTGDH tuples with the same Y value by choosing $a, b_1, b_3 \xleftarrow{R} \mathbb{Z}_p$, and generating

$$\begin{aligned}(X^a g^{b_1}, Y, Z g^{b_3}, (D_{XY})^a Y^{b_1}, (D_{XZ})^a X^{ab_3} Z^{b_1} g^{b_1 b_3},\\ D_{YZ} Y^{b_3}, W^a (D_{XY})^{ab_3} (D_{YZ})^{b_1} Y^{b_1 b_3}).\end{aligned}$$

Steiner *et al.* [45] show that polynomial indistinguishability of DDH implies polynomial indistinguishability of DTGDH, and from their proof one can see that for $t' = t + O(t_{\mathsf{exp}})$,

$$\mathsf{Adv}_G^{\mathrm{DTGDH}}(t) \le 3\mathsf{Adv}_G^{\mathrm{DDH}}(t').$$

VI. J-PAKE Security

Here we prove that the J-PAKE protocol is secure, in the sense that an adversary attacking the system cannot determine session keys with advantage non-negligibly greater than that of an online dictionary attack.

Theorem VI.1. *Let P be the protocol described in Fig. 1, using group G, and with a password dictionary of size N. Fix an adversary $\mathcal{A}$ that runs in time t, and makes n_{se} queries of type* Send *to different instances, and makes n_{ex}, n_{re}, n_{te}, n_{co} queries of type* Execute, Reveal, Test, Corrupt, *respectively. Then for $t' = O(t + (n_{\mathrm{se}} + n_{\mathrm{ex}} + n_{\mathrm{co}})t_{\mathsf{exp}})$:*

$$\begin{aligned}\mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) = \frac{n_{\mathrm{se}}}{N} + O\Bigg(\frac{(n_{\mathrm{se}} + n_{\mathrm{ex}})^2}{p} + n_{\mathrm{se}}\mathsf{Adv}_G^{\mathrm{DL}}(t') +\\ n_{\mathrm{se}}\mathsf{Adv}_G^{\mathrm{DSDH}}(t') + (n_{\mathrm{ex}} + n_{\mathrm{se}}^2)\mathsf{Adv}_G^{\mathrm{DDH}}(t') +\\ \mathsf{Adv}_{\mathsf{NIZK}}^{\mathrm{uzk}}(t', 2n_{\mathrm{se}} + n_{\mathrm{ex}}) + \mathsf{Adv}_{\mathsf{NIZK}}^{\mathrm{ext}}(t', 2n_{\mathrm{se}} + n_{\mathrm{ex}}) +\\ (n_{\mathrm{re}} + n_{\mathrm{te}})\mathsf{Adv}_{\mathrm{rExt}}^{\text{comp-ext}}(t')\Bigg),\end{aligned}$$

where $\mathsf{Adv}^{\mathrm{uzk}}$ and $\mathsf{Adv}^{\mathrm{ext}}$ are advantages for the security of the SE-NIZK, formally defined in Section VIII.

Proof: Our proof will proceed by introducing a series of protocols $P_0, P_1, \ldots, P_8$ related to P, with $P_0 = P$. We will bound the decrease in the advantage of $\mathcal{A}$ in each successive protocol, and finally in P_8, $\mathcal{A}$ will be reduced to a simple online guessing attack that will admit a straightforward analysis. We describe these protocols informally in Fig. 3.

We will assume the session key is simply the K value instead of $H(K)$, and use that for all Reveal and Test queries. Since $H(K)$ can be computed from K, this only strengthens our proof.

To simplify our analysis, we use the following terminology and shorthand. When we say an instance receives a value that has a corresponding NIZK proof, this implies that the instance was ready to accept that value and that the NIZK proof was valid. Otherwise, the instance simply rejects. We say a client (resp. server) instance is a *matching instance* if it has a *buddy* instance with the same X_1, X_2, X_3, X_4 values and also the same β (resp. α) value. We say a client (resp. server) instance is a *swapping instance* if it has a *buddy* instance with the same X_3, X_4, β (resp. X_1, X_2, α) values, but flipped X_1, X_2 (resp. X_3, X_4) values We say a client (resp. server) instance is an *almost matching/swapping instance* if it is a matching/swapping instance except for the value of X_4 (resp. X_2). We say an instance is *detached* if it is not a matching/swapping or almost matching/swapping instance.

For a detached instance, we say a Send query containing α (resp. β) to corresponds to a password pw if for the X_1, X_2, X_3, X_4 values of the instance receiving the Send query, $\mathsf{pw} = \mathrm{DL}(\alpha, X_1X_3X_4)/\mathrm{DL}(X_2, g)$ (resp. $\mathsf{pw} = \mathrm{DL}(\beta, X_1X_2X_3)/\mathrm{DL}(X_4, g)$). The exact way in which these values are computed depends on the specific protocol and will be explained later in the proof. For a matching/swapping instance, we say the Send query containing α or β always

P_0 **Original:** The original protocol P.

P_1 **Simulate and Extract:** Choose all passwords pw of clients randomly (and independently) and simulate all $\mathsf{Execute}$, Send, Reveal, Test, and $\mathsf{Corrupt}$ queries just like normal instances, except using the simulator for the NIZK proof in $\mathsf{Execute}$ and Send queries, and running the extractor for the NIZK proofs produced by the adversary, failing if the extraction fails.

P_2 **Force Uniqueness:** If any instance chooses an X_1, X_2, X_3, or X_4 value seen previously in the execution of the protocol, the protocol halts and the adversary fails.

P_3 **Disallow Trivial DL Attacks:** Take two instances Π_i^A and Π_j^B that generate X_1, X_2 and X_3, X_4, respectively. Then on any of the following situations, the protocol halts and the adversary fails.

1) Before sending a β query to Π_i^A, the adversary sends X_1', X_2' to Π_j^B where $X_1X_2 = X_1'X_2'$ but $X_1' \neq X_1$ and $X_1' \neq X_2$.
2) Before sending an α query to Π_j^B, the adversary sends X_3', X_4' to Π_i^A where $X_3X_4 = X_3'X_4'$ but $X_3' \neq X_3$ and $X_3' \neq X_4$.
3) Before sending an α query to Π_j^B, the adversary sends X_3', X_4' to Π_i^A and X_1' to Π_j^B, where $X_1X_3'X_4' = X_1'X_3X_4$, but $X_1' \neq X_1$.
4) Before sending a β query to Π_i^A, the adversary sends X_1', X_2' to Π_j^B and X_3' to Π_i^A, where $X_1X_2X_3' = X_1'X_2'X_3$, but $X_3' \neq X_3$.

P_4 **Check Password Guesses:** If before a $\mathsf{Corrupt}$ query, the adversary makes a Send query to an instance corresponding to a correct password pw using an α or β value such that the instance is not a matching instance or swapping instance, the protocol halts and the adversary succeeds.

P_5 **Randomize Session Keys For Wrong Password Guesses:** In any instance that is not a matching or swapping instance, and whose Send query (for α or β) corresponds to an incorrect password, set K randomly.

P_6 **Randomize Session Keys For Paired Instances:** In any matching or swapping instance, set K randomly (with matching buddies getting the same K).

P_7 **Randomize α and β:** If a $\mathsf{Corrupt}$ query has not occurred, generate α and β values randomly in all instances. For an instance that is not a matching nor swapping instance, and whose Send query (for α or β) corresponds to the correct password (i.e., after a $\mathsf{Corrupt}$ query), compute K as the other party would have.

P_8 **Randomize sk:** In any instance in which K is set randomly, set sk randomly (with matching buddies getting the same sk).

Fig. 3. Description of protocols P_0 through P_8

corresponds to the correct password pw. For an almost matching/swapping instance, we say the Send query containing α or β always corresponds to an incorrect password. Note that in P_8, the passwords aren't used anymore in any simulations, except for checking passwords in P_4. It is easy to see that this is equivalent to a protocol with a password guessing oracle, that is queried at most once per non-matching instance (since we know the discrete logs of all values). Then it is trivial to show that when the passwords are chosen randomly (and independently) from a set of size N the probability of an attacker succeeding is at most n_{se}/N.

For each i from 1 to 8, we will bound the increase in the advantage of $\mathcal{A}$ attacking protocol P_{i-1} compared to the advantage of $\mathcal{A}$ attacking protocol P_i.

Protocol P_1 (simulate and extract). In this protocol, we choose all passwords pw of clients randomly (and independently) and simulate all $\mathsf{Execute}$, Send, Reveal, Test, and $\mathsf{Corrupt}$ queries just like normal instances, except using the simulator for the NIZK proof in $\mathsf{Execute}$ and Send queries, and running the extractor for the NIZK proofs produced by the adversary, failing if the extraction fails.

Protocol P_1 behaves exactly like P_0, except for the simulation of the NIZK proofs and failing on unsuccessful NIZK extractions. From the unbounded zero-knowledge property, and the simulation-sound extractability property of the NIZK,

$$\mathsf{Adv}^{\text{ake}}_{P_0}(\mathcal{A}) = \mathsf{Adv}^{\text{ake}}_{P_1}(\mathcal{A}) + O\big(\mathsf{Adv}^{\text{uzk}}_{\mathsf{NIZK}}(t', 2n_{\text{se}} + n_{\text{ex}}) + \mathsf{Adv}^{\text{ext}}_{\mathsf{NIZK}}(t', 2n_{\text{se}} + n_{\text{ex}})\big)$$

Note that in P_1, the simulator obtains all adversarial values of $x_1, x_2, x_3, x_4, \mathsf{pw}$ from the NIZK extractions.

Protocol P_2 (force uniqueness). If any instance chooses an X_1, X_2, X_3, or X_4 value seen previously in the execution of the protocol, the protocol halts and the adversary fails.

Protocols P_1 and P_2 are only distinguishable if we hit the "birthday paradox".

$$\mathsf{Adv}^{\text{ake}}_{P_1}(\mathcal{A}) \quad = \quad \mathsf{Adv}^{\text{ake}}_{P_2}(\mathcal{A}) + O\left(\frac{(n_{\text{se}} + n_{\text{ex}})^2}{p}\right).$$

Protocol P_3 (disallow trivial DL attacks). Take two instances Π_i^A and Π_j^B that generate X_1, X_2 and X_3, X_4, respectively. Then on any of the following situations, Protocol P_3 halts and the adversary fails.

1) Before sending a β query to Π_i^A, the adversary sends X_1', X_2' to Π_j^B where $X_1X_2 = X_1'X_2'$ but $X_1' \neq X_1$ and $X_1' \neq X_2$.
2) Before sending an α query to Π_j^B, the adversary sends

X'_3, X'_4 to Π_i^A where $X_3X_4 = X'_3X'_4$ but $X'_3 \neq X_3$ and $X'_3 \neq X_4$.

3) Before sending an α query to Π_j^B, the adversary sends X'_3, X'_4 to Π_i^A and X'_1 to Π_j^B, where $X_1X'_3X'_4 = X'_1X_3X_4$, but $X'_1 \neq X_1$.
4) Before sending a β query to Π_i^A, the adversary sends X'_1, X'_2 to Π_j^B and X'_3 to Π_i^A, where $X_1X_2X'_3 = X'_1X'_2X_3$, but $X'_3 \neq X_3$.

We will bound the advantage increase in each part using reduction from DL. Take an instance of DL with value X.

For part 1, choose a random client instance Π_i^A and set $X_1 = X$ and $X_2 = g^{x_2}$. Then if the adversary sends X'_1, X'_2 to an instance of B where $X_1X_2 = X'_1X'_2$ but $X'_1 \neq X_1$ and $X'_1 \neq X_2$, use x'_1, x'_2 obtained from extracting witness from NIZK proofs (or from the simulation of an instance of A), and compute the discrete log of X as $x'_1x'_2/x_2$. Note that Π_i^A can still be simulated perfectly since the discrete log of X_1 is never used except in an NIZK proof, which is already simulated.

Part 2 is similar.

For part 3, choose a random client instance Π_i^A and set $X_1 = X$ and $X_2 = g^{x_2}$. Then if the adversary sends X'_3, X'_4 to Π_i^A and X'_1, X'_2 (for some X'_2) to an instance Π_j^B that produced X_3, X_4, where $X_1X'_3X'_4 = X'_1X_3X_4$ but $X'_1 \neq X_1$, use x'_1, x'_3, x'_4 obtained from extracting witnesses from NIZK proofs (or from simulations of instances of A and B), and compute the discrete log of X as $x'_1x_3x_4/x'_3x'_4$. Note that Π_i^A can still be simulated perfectly since the discrete log of X_1 is never used except in an NIZK proof, which is already simulated.

Part 4 is similar.

$$\mathsf{Adv}^{\mathrm{ake}}_{P_2}(\mathcal{A}) = \mathsf{Adv}^{\mathrm{ake}}_{P_3}(\mathcal{A}) + O\left(n_{\mathrm{se}}\mathsf{Adv}^{\mathrm{DL}}_{G}(t')\right).$$

Protocol P_4 (check password guesses). If before a Corrupt query, the adversary makes a Send query to an instance corresponding to a correct password pw using an α or β value such that the instance is not a matching instance or swapping instance, the protocol halts and the adversary succeeds.

Note that to determine if a Send query corresponds to a correct password pw, it is only necessary to know either the 2 discrete logs (of α and X_2, or β and X_4) or one of the discrete logs and the password pw. For instance, with just pw and the discrete log y of α we can check if $g^{y/\mathsf{pw}} = X_2$. Also note that for an NIZK proof generated by the adversary, this value y can be extracted by the NIZK extractor and verified. In every protocol and reduction argument used before P_8, we will know pw, and at least one of the discrete logs for every Send query of interest, and in P_8, we will know all discrete logs for any Send query of interest.

Note that in P_4, the advantage of the adversary can only increase.

$$\mathsf{Adv}^{\mathrm{ake}}_{P_3}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ake}}_{P_4}(\mathcal{A}).$$

Protocol P_5 (randomize session keys for wrong password guesses). In any instance that is not a matching or swapping instance, and whose Send query (for α or β) corresponds to an incorrect password, we set K randomly.

At this point we may assume that for a Send query with α or β, the recipient is either matching or swapping, the query corresponds to an incorrect password guess, or the query corresponds to a correct password guess and either there has been a Corrupt query or the simulation is halted.

We split this proof up and first examine a protocol $P_{4.5}$ that only sets K randomly in the relevant instances of A. We bound the advantage increase using a reduction from DSDH.

Use a hybrid argument and consider the ith instance of A, where all previous instances set K randomly if necessary, and all later instances use the normally computed K.

Given a DSDH instance (X, Y), let Π_i^A set $X_1 = g^{x_1}$ and $X_2 = X$. When Π_i^A receives X'_3, X'_4 values, use x'_3, x'_4 values obtained from extracting witnesses from NIZK proofs (or from the simulation of an instance of B), and set $\alpha = (X^{x_1+x'_3+x'_4})^{\mathsf{pw}}$. If that instance receives a Send query with β, use the y value obtained from extracting a witness for β (or from the simulation of an instance of B), and let $\mathsf{pw}' = y/x_4$ be the password associated with the β value and compute $K = X^{(x_1+x'_3)x'_4\mathsf{pw}'}Y^{x'_4(\mathsf{pw}'-\mathsf{pw})}$. Note that if x is the (unknown) discrete log of X, then K would have been computed by A as $K = (\beta g^{-x'_4x\mathsf{pw}})^x = g^{x(x_1+x'_3)x'_4\mathsf{pw}'}g^{x^2x'_4(\mathsf{pw}'-\mathsf{pw})}$, which is the actual K value computed if $Y = g^{x^2}$, or if $\mathsf{pw}' = \mathsf{pw}$ (which will always be the case for matching or swapping instances). On the other hand, if Y is random and this is an incorrect password guess (i.e., $\mathsf{pw}' \neq \mathsf{pw}$), then K will be random since it will depend on Y, since $x'_4 \neq 0$ (because the protocol checks that $X_4 \neq 1$) and $\mathsf{pw}' - \mathsf{pw} \neq 0$.

All that is left to show is that any instance Π_j^B that generates X_3, X_4, and has received X'_1, X'_2, and that that receives a Send query using this α may be simulated. If it doesn't use this α, then it can obtain the discrete log of α and satisfy P_4. If it uses this α, and $X'_2 \neq X_2$, then it can obtain the discrete log of X'_2 and satisfy P_4. If it uses this α and $X'_2 = X_2$, then because the NIZK proof for α specifies $X_1X'_3X'_4$, we can assume $X_1X'_3X'_4 = X_1X_3X_4$. But by P_3, this implies $X'_1 = X_1$. But then $X'_3X'_4 = X_3X_4$, and again by P_3, this implies $(X'_3, X'_4) = (X_3, X_4)$ or $(X'_3, X'_4) = (X_4, X_3)$, i.e., it is matching or swapping. Therefore no password test is necessary, so no discrete log is necessary.

The number of hybrids is bounded by the number of different instances, which is bounded by n_{se}. Therefore we get the following bound.

$$\mathsf{Adv}^{\mathrm{ake}}_{P_4}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ake}}_{P_{4.5}}(\mathcal{A}) + n_{\mathrm{se}}\mathsf{Adv}^{\mathrm{DSDH}}_{G}(t').$$

Using a perfectly analogous argument, we can prove the same bound on the advantage increase from P_5 to $P_{4.5}$. Combining the two results above, we have

$$\mathsf{Adv}^{\mathrm{ake}}_{P_4}(\mathcal{A}) = \mathsf{Adv}^{\mathrm{ake}}_{P_5}(\mathcal{A}) + O(n_{\mathrm{se}}\mathsf{Adv}^{\mathrm{DSDH}}_{G}(t')).$$

Protocol P_6 (randomize session keys for paired instances). In any matching or swapping instance, we set K randomly (with matching buddies getting the same K).

We bound the advantage increase in the advantage of the adversary from P_6 to P_5 using a reduction from DTGDH.

Use a hybrid argument and consider the ith instance of A (Π_i^A) and jth instance of B (Π_j^B), where all lexicographically previous instances of A and B that are full-matching or full-swapping set K randomly, and all lexicographically later instances set K normally. We will proceed assuming Send queries between these instances. For Execute queries, the results hold by assuming they are matching Send queries.

Take a DTGDH instance $(X, Y, Z, D_{XY}, D_{XZ}, D_{YZ}, W)$. For Π_i^A, set $X_1 = g^{x_1}$ and $X_2 = X$, and for Π_j^B, set $X_3 = Y$, $X_4 = Z$. On a query (X_3', X_4') to Π_i^A with $\{X_3', X_4'\} = \{X_3, X_4\}$, set $\alpha = (D_{XY} D_{XZ} X^{x_1})^{\mathsf{pw}}$. For other queries to A, compute α using either the x_3' and x_4' values obtained by extraction from the NIZK proof, or from the simulation of an instance of B, or the D_{XY} and D_{XZ} values. Similarly, on a query (X_1', X_2') to Π_j^B with $\{X_1', X_2'\} = \{X_1, X_2\}$, set $\beta = (D_{XZ} D_{YZ} Z^{x_1})^{\mathsf{pw}}$. For other queries to B, compute β using either the x_1' and x_2' values obtained by extraction from the NIZK proof, or from the simulation of an instance of A, or the D_{XZ} and D_{YZ} values.

Now consider a Send query to Π_i^A with a value β. If this is a matching or swapping instance with buddy Π_j^B, set $K = (D_{XZ}^{x_1} W)^{\mathsf{pw}}$. If $W = TGDH(X, Y, Z)$, then this simulates computing K normally, and if W is random, then K is random.

Now for the other cases, if β is not from Π_j^B, then we can extract the discrete log, and satisfy P_4. If β is taken from Π_j^B, $X_3' = X_3$ and $\{X_1', X_2'\} = \{X_1, X_2\}$ for the values X_1', X_2' received by Π_j^B, by P_3. But because this is not matching of swapping, $X_4' \neq X_4$, so the instance is almost matching/swapping and the corresponding password guess would be incorrect. Again this satisfies P_4. Note that in the case when the password guess is correct and we are continuing (which because of P_4 means there was a Corrupt query), then we will know $\mathrm{DL}(X_4, g)$, so we can compute K as B would.

Now consider a Send query to Π_i^B with a value α. If this is a matching or swapping instance with buddy Π_i^A, set $K = (D_{XZ}^{x_1} W)^{\mathsf{pw}}$. If $W = TGDH(X, Y, Z)$, then this simulates computing K normally, and if W is random, then K is random.

Now for the other cases, if α is not from Π_i^A, then we can extract the discrete log, and satisfy P_4. If α is taken from Π_i^A, $X_1' = X_1$ and $\{X_3', X_4'\} = \{X_3, X_4\}$ for the values X_3', X_4' received by Π_i^A, by P_3. But because this is not matching of swapping, $X_2' \neq X_2$, so the instance is almost matching/swapping and the corresponding password guess would be incorrect. Again this satisfies P_4. Note that in the case when the password guess is correct and we are continuing (which because of P_4 means there was a Corrupt query), then we will know $\mathrm{DL}(X_2, g)$, so we can compute K as A would.

Because there are at most n_{se}^2 pairs of instances receiving Send queries, and at most n_{ex} pairs of instances in execute queries, we get the following bound.

$$\mathsf{Adv}^{\mathrm{ake}}_{P_5}(\mathcal{A}) = \mathsf{Adv}^{\mathrm{ake}}_{P_6}(\mathcal{A}) + O((n_{\mathrm{ex}} + n_{\mathrm{se}}^2)\mathsf{Adv}^{\mathrm{DTGDH}}_{G}(t')).$$

Protocol P_7 (randomize α and β). If a Corrupt query has not occurred, we generate α and β values randomly in all instances. For an instance that is not a matching nor swapping instance, and whose Send query (for α or β) corresponds to the correct password (i.e., after a Corrupt query), compute K as the other party would have.

First consider generating only the α values randomly.

Use a hybrid argument and consider the ith instance of A (Π_i^A), where all previous instances use random exponents in place of $x_2\mathsf{pw}$ to compute their α values and all later instances use actual α values.

We reduce from DDH. If the attacker succeeds in the protocol, we let the DDH attacker guess $b = 1$ (i.e., a valid DDH instance). Take a DDH instance (X, Y, Z). Let Π_i^A set $X_1 = X$, $X_2 = Y$. On a Send query with X_3', X_4', use x_3', x_4' values obtained from the NIZK extractor, or from a simulation of B, and compute $\alpha = (Z X_2^{x_3' + x_4'})^{\mathsf{pw}}$. On a Send query with β, use the y value obtained from the NIZK extractor to satisfy P_7, and complete the simulation of A. For other instances of A which receive Send queries with β corresponding to the correct password, use the x_3', x_4', y values obtained from the NIZK extractor, or from the simulation of B to compute K as B would. This is critical when α is random, to ensure that the instance computes the same K as the adversary would. Given a real DDH tuple, this will be equivalent to P_6. Given a random tuple, this will be equivalent to the hybrid protocol. If the adversary succeeds in the protocol, output $b = 1$, otherwise output $b = 0$.

To show that it is equivalent, we must show that we can determine correct password guesses for Send queries that do not correspond to matching, swapping, or almost matching/swapping instances. Consider a Send query to Π_j^B with a value α, where Π_j^B is not matching, swapping, or almost matching/swapping. If α is not from Π_i^A, then we can extract the discrete log, determine whether it is correct or not and satisfy P_4. Note that if the password guess is correct and we are continuing (which because of P_4 means there was a Corrupt query), then we will know $\mathrm{DL}(X_2, g)$, so we can compute K as A would. If α is taken from Π_i^A, then $X_1' = X_1$ and $\{X_3', X_4'\} = \{X_3, X_4\}$ for the values X_3', X_4' received by Π_i^A, by P_3. So Π_j^B is a matching/swapping or almost matching/swapping instance, and we set K randomly in either case, by P_5 and P_6.

The reduction for generating random β values is analogous.

$$\mathsf{Adv}^{\mathrm{ake}}_{P_6}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ake}}_{P_7}(\mathcal{A}) + 2n_{\mathrm{se}}\mathsf{Adv}^{\mathrm{DDH}}_{G}(t').$$

Protocol P_8 (randomize sk). In any instance in which K is set randomly, we set sk randomly (with matching buddies getting the same sk).

This protocol is computationally indistinguishable from the previous one, as rExt is a computational randomness extractor.

$$\mathsf{Adv}^{\mathrm{ake}}_{P_7}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ake}}_{P_8}(\mathcal{A}) + (n_{\mathrm{re}} + n_{\mathrm{te}})\mathsf{Adv}^{\text{comp-ext}}_{\mathrm{rExt}}(t').$$

That concludes the proof. ∎

VII. J-PAKE Label Variants

In the proof of Theorem VI.1, the labels in the NIZK proofs are the ones suggested in [24], namely, the label is the identity of the principal who generated the proof. Since principals are either server or client, this label implicitly indicates whether the proof has been generated by a client or a server.

In the OpenSSL implementation [39] of J-PAKE (more precisely, in programs `s_client` and `s_server`) and in the Firefox Sync [16] implementation, the label is just "client" or "sender" for the client and "server" or "receiver" for the server. We could actually adapt our proof to these labels, in the case where there is only a single server. However, when there are many servers, the adversary can make a client A authenticate to a server B_1, while making the client believe it authenticates to another server B_2, if A uses the same password on B_1 and B_2. The attack just consists in redirecting flows from A to B_1 to B_2.

Finally, if labels contained more information, such as (A, X_1, X_2) for π_1 and π_2, $(A, B, X_1, X_2, X_3, X_4)$ for π_α, and the corresponding data for the server's labels, then the proof would be simpler and tighter. In particular, protocol P_3 would not be necessary anymore, and we could use random self-reducibility of the DSDH assumption. If, in addition, the server B only sends its first flow after receiving the client's first flow, then the proof would be even simpler and tighter, by using partial random self-reducibility of the DTGDH assumption. With these changes,

Theorem VII.1. *Let P be the protocol described in Fig. 1 with modified labels as described above, using group G, and with a password dictionary of size N, and where the first flow of the server is sent after receiving the first flow of the client. Fix an adversary $\mathcal{A}$ that runs in time t, and makes n_{se} queries of type* Send *to different instances, and makes n_{ex}, n_{re}, n_{te}, n_{co} queries of type* Execute, Reveal, Test, Corrupt, *respectively. $t' = O(t + (n_{\mathrm{se}} + n_{\mathrm{ex}} + n_{\mathrm{co}})t_{\mathsf{exp}})$:*

$$\mathsf{Adv}^{\mathrm{ake}}_{P}(\mathcal{A}) = \frac{n_{\mathrm{se}}}{N} + O\Bigg(\frac{(n_{\mathrm{se}} + n_{\mathrm{ex}})^2}{p} + \mathsf{Adv}^{\mathrm{DSDH}}_{G}(t') + (n_{\mathrm{se}} + 1)\mathsf{Adv}^{\mathrm{DDH}}_{G}(t') + \mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{NIZK}}(t', 2n_{\mathrm{se}} + n_{\mathrm{ex}}) + \mathsf{Adv}^{\mathrm{ext}}_{\mathsf{NIZK}}(t', 2n_{\mathrm{se}} + n_{\mathrm{ex}}) + (n_{\mathrm{re}} + n_{\mathrm{te}})\mathsf{Adv}^{\text{comp-ext}}_{\mathrm{rExt}}(t')\Bigg).$$

Finally, we note that if we set $sk = H(K)$ where H is a random oracle (or, in other words, if the randomness extractor rExt is a random oracle), we could replace the DSDH assumption with the CSDH assumption, and specifically replace $\mathsf{Adv}^{\mathrm{DSDH}}_{G}(t')$ with $n_{\mathrm{ro}}\mathsf{Adv}^{\mathrm{CSDH}}_{G}(t')$ in the theorem above, where n_{ro} is the number of random oracle queries. This follows from modifying the reduction in P_5 to use the adversaries random oracle queries to H to determine guesses for the square Diffie-Hellman values.

VIII. Simulation-Sound Extractable Non-Interactive Zero-Knowledge Proofs

Before explaining how to extend the proof of security of J-PAKE to use Schnorr proofs instead of SE-NIZK proofs, let us first formally recall some definitions from [22], extended to the case of labeled non-interactive proof systems.

A. Non-interactive proof systems

Intuitively a proof system is a protocol which enables a prover to prove to a verifier that a given word or statement x is in a given NP-language. We are interested in non-interactive proofs, i.e., proofs such that the prover just sends one message.

More formally, let $\mathscr{L}$ be a language in NP with witness relation $\mathcal{R}$, i.e., $\mathscr{L} = \{x \mid \exists\omega,\ \mathcal{R}(x,\omega) = 1\}$. We suppose $\mathcal{R}(x,\omega)$ can be checked in poly-time. A labeled non-interactive proof system for $\mathscr{L}$ is defined by a tuple $(\mathrm{Setup}, \mathsf{PK}, \mathsf{VK})$, such that:

$\mathrm{Setup}(1^\kappa)$ outputs a common reference string (CRS) σ;

$\mathsf{PK}(\sigma, x, \omega, \ell)$ takes as input a CRS $\sigma \xleftarrow{R} \mathrm{Setup}(1^\kappa)$, a word $x \in \mathscr{L}$, a witness ω for x (such that $\mathcal{R}(x,\omega) = 1$), and a label $\ell \in \{0,1\}^*$, and outputs a proof π that x is in $\mathscr{L}$, for label ℓ;

$\mathsf{VK}(\sigma, x, \pi, \ell)$ takes as input the CRS σ, a word x, a proof π, and a label ℓ, and outputs 1 to indicate acceptance and 0 otherwise;

and such that the following two properties hold:

Perfect completeness. A non-interactive proof is complete if an honest prover knowing a statement $x \in \mathscr{L}$ and a witness ω for x can convince an honest verifier that x is in $\mathscr{L}$, for any label. More formally, $(\mathrm{Setup}, \mathsf{PK}, \mathsf{VK})$ is said to be perfectly complete, if for all ℓ, for all $x \in \mathscr{L}$ and ω such that $\mathcal{R}(x,\omega) = 1$, for all $\sigma \xleftarrow{R} \mathrm{Setup}(1^\kappa)$, we have $\mathsf{VK}(\sigma, x, \mathsf{PK}(\sigma, x, \omega, \ell), \ell) = 1$;

Soundness. A non-interactive proof is sound if no poly-time adversary $\mathcal{A}$ can prove a false statement with non-negligible probability. Let $\mathsf{Adv}^{\mathrm{sound}}_{\mathsf{NIZK}}(\mathcal{A})$ be:

$$\Pr\Big[\sigma \xleftarrow{R} \mathrm{Setup}(1^\kappa);\ (\ell, x, \pi) \xleftarrow{R} \mathcal{A}(\sigma) : \mathsf{VK}(\sigma, x, \pi, \ell) = 1 \text{ and } x \notin \mathscr{L}\Big] \leq \varepsilon.$$

B. Simulation-Sound Extractable Non-interactive zero-knowledge proofs (SE-NIZK)

An (unbounded) **SE-NIZK** (simulation-sound extractable non-interactive zero-knowledge) proof is a non-interactive proof system with two simulators Sim_1 and Sim_2, which can simulate Setup and PK, but such that Sim_2 does not need any witness. More formally an SE-NIZK proof is defined by

a tuple $(\mathrm{Setup}, \mathsf{PK}, \mathsf{VK}, \mathrm{Sim}_1, \mathrm{Sim}_2, \mathrm{Ext})$ such that $(\mathrm{Setup}, \mathsf{PK}, \mathsf{VK})$ is a non-interactive proof system, and:

$\mathrm{Sim}_1(1^\kappa)$ generates a CRS σ and two trapdoors τ, ξ, such that Sim_2 can use τ to simulate proofs under σ, and Ext can use ξ to extract witnesses from proofs;

$\mathrm{Sim}_2(\sigma, \tau, x, \ell)$ takes as input the CRS σ, the corresponding trapdoor τ, a word x (not necessarily in $\mathscr{L}$), and a label ℓ, and outputs a (fake or simulated) proof π for x;

$\mathrm{Ext}(\sigma, \xi, x, \pi, \ell)$ takes as input the CRS σ, the corresponding trapdoor ξ, a word x, a label ℓ, and a valid proof π, and extracts from π a witness ω for x if possible (and otherwise outputs $\perp$).

and such that it verifies the following property:

Unbounded zero-knowledge. An NIZK proof is (unbounded) zero-knowledge if simulated proofs are indistinguishable from real proofs. Let $\mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{NIZK}}(\mathcal{A})$ be:

$$\Pr\left[\sigma \xleftarrow{R} \mathrm{Setup}(1^\kappa) : \mathcal{A}^{\mathsf{PK}(\sigma,\cdot,\cdot,\cdot)}(\sigma) = 1\right] - \Pr\left[(\sigma,\tau,\xi) \xleftarrow{R} \mathrm{Sim}_1(1^\kappa) : \mathcal{A}^{\mathrm{Sim}'(\sigma,\tau,\cdot,\cdot,\cdot)}(\sigma) = 1\right]$$

where $\mathrm{Sim}'(\sigma, \tau, x, \omega, \ell) = \mathrm{Sim}_2(\sigma, \tau, x, \ell)$; we write $\mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{NIZK}}(t, n_{\mathrm{sim}}) = \max_{\mathcal{A}} \{\mathsf{Adv}(\mathcal{A})\}$, where the maximum is taken over all adversaries of time complexity at most t and making at most n_{sim} queries to Sim' or PK.

Simulation-sound extractability. An NIZK proof is simulation-sound extractable if Ext can extract a witness from any proof generated by the adversary, even if the adversary can see simulated proofs. Let $\mathsf{Adv}^{\mathrm{ext}}_{\mathsf{NIZK}}(\mathcal{A})$ be:

$$\Pr\Big[(\sigma,\tau,\xi) \xleftarrow{R} \mathrm{Sim}_1(1^\kappa); (x,\pi) \xleftarrow{R} \mathcal{A}^{\mathrm{Sim}_2(\sigma,\tau,\cdot,\cdot)}(\sigma,\xi) : \mathsf{VK}(\sigma,x,\pi,\ell) = 1,\ ((x,\ell),\pi) \notin S \text{ and } \mathcal{R}(x, \mathrm{Ext}(\sigma,\xi,x,\pi,\ell)) = 0\Big]$$

where S is the set of query-response pairs $((x, \ell), \pi)$ for $\mathrm{Sim}_2(\sigma, \tau, \cdot, \cdot)$; we write $\mathsf{Adv}^{\mathrm{ext}}_{\mathsf{NIZK}}(t, n_{\mathrm{sim}}) = \max_{\mathcal{A}} \{\mathsf{Adv}(\mathcal{A})\}$, where the maximum is taken over all adversaries of time complexity at most t and making at most n_{sim} queries to Sim_2.

IX. Algebraic Simulation-Sound Extractable NIZK and Schnorr Proofs

Schnorr signatures [43] can be seen as the most efficient NIZK proof (in the random-oracle model) for the discrete logarithm language. However, while there exists an extractor using rewinding for Schnorr signatures using the forking lemma [42], no straight-line extractor (which can directly extract the witness from the proof, as in our definition in Section II-B) is known. Unfortunately, using rewinding extractors would induce an exponential blow-up in our security reduction for J-PAKE (as it is the case in the proof for blind signatures in [42]).

In this section, we show how to circumvent this problem if we assume the adversary is algebraic. After recalling the definition of algebraic adversaries, we first introduce the notion of algebraic-simulation-sound extractable NIZK (which is a weak variant of simulation-sound extractable NIZK). We then show that Schnorr proofs are algebraic-simulation-sound extractable NIZK proofs (in the random-oracle model), and that the J-PAKE proof can be adapted to only require that the NIZK proofs are algebraic-simulation-sound extractable instead of simulation-sound extractable.

A. Algebraic Adversaries

Algebraic adversaries were introduced by Paillier and Vergnaud in [40]. An adversary is algebraic (with respect to G) if it is limited to perform group operations on group elements in G, and no other operations. Operations on non-group elements are not restricted.

This means that if the adversary only see groups elements $g_1, \ldots, g_n$, and if it outputs some group element h, then there is a way to extract from the adversary $\lambda_1, \ldots, \lambda_n \in \mathbb{Z}_p^n$ such that:

$$h = g_1^{\lambda_1} \cdots g_n^{\lambda_n}.$$

We call $(\lambda_1, \ldots, \lambda_n)$ the discrete logarithms of h in base $(g_1, \ldots, g_n)$. This property even holds for group elements given as inputs to the random oracle, if any.

Assuming the adversary to be algebraic is a weaker assumption than using the (non-programmable) generic group model [44]. In particular, all the proofs of this section are also valid in the non-programmable generic group model.

A subtle point in all this section is that that the "adversary" for the extractor Ext (or what generates the proof π) is not simply the adversary of J-PAKE, but very roughly, a "part of" the reduction from J-PAKE to some hard problem. This means that we also need the reduction to be algebraic and we cannot assume that the reduction knows the discrete logarithms of $(g_2, \ldots, g_n)$ in base g_1. Indeed, these values $(g_1, \ldots, g_n)$ will often come from an instance of a hard problem such as DDH or DTGDH.

A completely wrong but very tempting way to make our proof work for Schnorr proofs would be to just forget about the bases and to say that in the generic group model, by controlling the group operation oracles[5], Ext can extract the discrete logarithm of any element. However, if that were true, NIZK proofs would be useless (since we can compute the discrete logarithm of any element without them) and could be removed. But, clearly, the J-PAKE protocol without NIZK proof is insecure.

What is going wrong is that this solution supposes that Ext has full control over the group operation oracles, but the reduction also needs control on these oracles to work properly. So there is a conflict for the group operation oracles. The introduction of bases for discrete logarithms is a way to solve

[5]In the generic group model, group element are represented by random strings, and group operations are performed by calling some group operation oracles.

this conflict. We remark there is no such conflict with the random oracle, because we suppose that the NIZK has full control on it, and that it is not the same random oracle as the one used in J-PAKE to derive the secret key sk.

B. Schnorr Proofs and Issues

Schnorr Proof for Discrete Logarithm. Let us recall the Schnorr proof for the discrete logarithm language (see Section II-B for notations). There is no setup and no CRS, but a random oracle[6] $H : \{0,1\}^* \to \mathbb{Z}_p$. There is no trapdoor τ and ξ: the trapdoor τ (for simulation Sim_2) consists in being able to program the random oracle and the trapdoor ξ (for extraction Ext) consists in being able to extract the discrete logarithms of all group elements. This input of the discrete logarithms of all group elements for Ext is implicit in what follows. When needed, we indicate the base which is used in exponent of Ext, e.g., $\mathrm{Ext}^{(g_1,\ldots,g_n)}(\sigma,\xi,x,\pi,\ell)$.

The Schnorr proof for the word $(u,h) = (h^r,h)$ and label ℓ, with witness $r = \omega \in \mathbb{Z}_p$ is $\pi = (z,e)$, with $r' \xleftarrow{R} \mathbb{Z}_p$, $u' \leftarrow h^{r'}$, $e \leftarrow H(\ell,(u,h),u')$ and $z \leftarrow r' - er \bmod p$. Checking a proof $\pi = (z,e)$ for (u,h) and ℓ consists in computing $u' \leftarrow h^z u^e$ and checking that $e = H(\ell,(u,h),u')$.

The simulator $\mathrm{Sim}_2(\sigma,\tau,x = (u,h),\pi,\ell)$ just picks e and z at random in $\mathbb{Z}_p$, and then programs H such that $H(\ell,x,u') = e$, with $u' = h^z u^e$. Unbounded zero-knowledge is straightforward: $\mathsf{Adv}^{\mathrm{uzk}} \le (n_{\mathrm{ro}} + n_{\mathrm{sim}})/p$, with n_{ro} the number of queries to the random oracle and n_{sim} the number of queries to Sim', since u' is uniform in $\mathbb{Z}_p$, and $H(\ell,x,u')$ is not already defined (and so can be programmed) with probability at least $1 - (n_{\mathrm{ro}} + n_{\mathrm{sim}})/p$.

Issues. Ideally, we would like to prove that Schnorr proofs are extractable when adversaries are algebraic, or in other words, when the extractor Ext also takes as input the discrete logarithms of all group elements of the proof and the word (in some base $(g_1,\ldots,g_n)$).

However, two issues arise. First, if the adversary knows the discrete logarithms of $g_2,\ldots,g_n$ in base g_1, and the extractor does not, there seems to be no way to extract the witness. For example, let us suppose that the base is (g_1,g_2,g_3) with $g_1 = g \xleftarrow{R} G$, $g_2 \leftarrow g^\alpha$ and $g_3 \leftarrow g^\beta$, with $\alpha,\beta \xleftarrow{R} \mathbb{Z}_p$. Then, if the adversary is able to compute α and β, it can pick $s,s' \xleftarrow{R} Z_p$, set $(u,h) \leftarrow (g_2^s, g)$, $u' \leftarrow g_3^{s'}$, $e \leftarrow H(\ell,(u,h),u')$, $z \leftarrow s'\beta - es\alpha \bmod p$, and $\pi \leftarrow (z,e)$. The fact the adversary is algebraic only enables us to recover s, s', and z, while the fact the proof is valid only ensures that $g^z = u'u^{-e} = g_2^{s'} g_3^{-se}$. Hence it is not clear at all how to extract a scalar r such that $u = g^r$.

Second, when doing a proof by games (or protocols as in Section VI), we use various assumptions, and so the base $(g_1,\ldots,g_n)$ (which basically depends on the assumption) changes during the proof. While this is transparent for the J-PAKE adversary, which never really sees $(g_1,\ldots,g_n)$ but derived elements from this base (the elements of the potential CRS and parameters, and the elements sent in the protocol), this is not the case for the extractor Ext which explicitly needs to be given the discrete logarithms of all group elements in the base $(g_1,\ldots,g_n)$. So we need to ensure that Ext does not change its behavior when we change bases. We will do it by introducing a property called *base indistinguishability*.

[6]This random oracle should be different from the one in the J-PAKE. We recall that using prefixes enable to generate multiple random oracles from one random oracle.

C. Hard-Linear Distributions and Algebraic Simulation-Sound Extractable NIZK

Let us first introduce the notion of a hard-linear distribution. A hard-linear distribution $\mathcal{D}$ is a distribution of tuples in G^n (for some n), such that, given G and $(g_1,\ldots,g_n) \xleftarrow{R} \mathcal{D}$, it is computationally hard to find $(\mu_1,\ldots,\mu_n) \neq 0^n$ such that $g_1^{\mu_1}\cdots g_n^{\mu_n} = 1$. More precisely, let $\mathsf{Adv}_{\mathcal{D}}^{\text{hard-lin}}(\mathcal{A})$ be

$$\Pr\left[(g_1,\ldots,g_n) \xleftarrow{R} \mathcal{D};\, (\mu_1,\ldots,\mu_n) \xleftarrow{R} \mathcal{A}(g_1,\ldots,g_n) : g_1^{\mu_1}\cdots g_n^{\mu_n} = 1\right].$$

We also suppose in hard-linear distributions that $g_1 = g$, a random generator of G (this can be done without loss of generality) and that there is a way to generate the base $(g_1,\ldots,g_n) \xleftarrow{R} \mathcal{D}$, such that the discrete logarithms $\nu_2,\ldots,\nu_n$ of $g_2,\ldots,g_n$ in base $g_1 = g$ are known.

Now to define algebraic-simulation-sound extractable NIZK, we replace the simulation-sound extractability property by the following two properties (where we assume for the sake of simplicity, that σ, τ, and ξ do not contain group elements):

Weak algebraic simulation-sound extractability. It is similar to simulation-sound extractability except the extractor is given the discrete logarithms of all group elements in a base $(g = g_1)$, with $g \xleftarrow{R} G \setminus \{1\}$ $(n = 1)$. Formally, for any poly-time algebraic adversary $\mathcal{A}$, let $\mathsf{Adv}_{\mathsf{NIZK}}^{\text{w-a-ext}}(\mathcal{A})$ be

$$\Pr\Big[(\sigma,\tau) \xleftarrow{R} \mathrm{Sim}_1(1^\kappa);\, g \xleftarrow{R} G;\\ (x,\pi) \xleftarrow{R} \mathcal{A}^{(g),\mathrm{Sim}_2(\sigma,\tau,\cdot,\cdot),\mathrm{Ext}^{(g)}(\sigma,\xi,\cdot,\cdot,\cdot)}(G,g,\sigma,\xi) :\\ \mathsf{VK}(\sigma,x,\pi) = 1,\ ((x,\ell),\pi) \notin S\\ \text{and } \mathcal{R}(x,\mathrm{Ext}^{(g)}(\sigma,\xi,x,\pi,\ell)) = 0\Big]$$

where S is the set of query-response pairs $((x,\ell),\pi)$ for $\mathrm{Sim}_2(\sigma,\tau,\cdot,\cdot)$.

Base indistinguishability. A NIZK proof is base-indistinguishable if everything is indistinguishable when used with the base $(g = g_1)$ with $g \xleftarrow{R} G \setminus \{1\}$, or with a hard linear base $(g_1,\ldots,g_n) \xleftarrow{R} \mathcal{D}$. More formally, for any poly-time algebraic adversary $\mathcal{A}$, let

$\mathsf{Adv}^{\text{base-ind}}_{\mathsf{NIZK},\mathcal{D},\mathcal{D}'}(\mathcal{A})$ be:

$$\Pr\Big[(\sigma,\tau)\xleftarrow{R}\mathrm{Sim}_1(1^\kappa);\,(g_1,\ldots,g_n)\xleftarrow{R}\mathcal{D}:\mathcal{A}^{\mathrm{Sim}_2(\sigma,\tau,\cdot,\cdot),\mathrm{Ext}^{(g_1)}(\sigma,\xi,\cdot,\cdot,\cdot)}(G,g_1,\ldots,g_n,\sigma)=1\Big]$$
$$-\Pr\Big[(\sigma,\tau)\xleftarrow{R}\mathrm{Sim}_1(1^\kappa);\,(g_1,\ldots,g_n)\xleftarrow{R}\mathcal{D}:\mathcal{A}^{\mathrm{Sim}_2(\sigma,\tau,\cdot,\cdot),\mathrm{Ext}^{(g_1,\ldots,g_n)}(\sigma,\xi,\cdot,\cdot,\cdot)}(G,g_1,\ldots,g_n,\sigma)=1\Big]$$

where in the first case, $(g_1,\ldots,g_n)$ are generated such that their discrete logarithms $1,\nu_2,\ldots,\nu_n$ in base g_1 are known, and so the discrete logarithm λ in base g_1 of any group element $v\in G$ generated by $\mathcal{A}$ (needed by $\mathrm{Ext}^{(g_1)}$) can be computed from the discrete logarithms $(\lambda_1,\ldots,\lambda_n)$ in base $(g_1,\ldots,g_n)$ of v (these discrete logarithms can be extracted from $\mathcal{A}$, since $\mathcal{A}$ is algebraic and only take as input the following group elements: $g_1,\ldots,g_n$): $\lambda=\lambda_1+\nu_2\lambda_2+\cdots+\nu_n\lambda_n$. In the second case, no such transform is required since $\mathrm{Ext}^{(g_1,\ldots,g_n)}$ just need the discrete logarithms in base $(g_1,\ldots,g_n)$, which can be directly extracted from the algebraic adversary $\mathcal{A}$. Ext is assumed to return $\perp$, when (x,π,ℓ) comes from a query-response pair of Sim_2.

The resulting scheme is called an *algebraic-simulation-sound extractable NIZK*. A priori, an algebraic-simulation-sound extractable is not an SE-NIZK, but an SE-NIZK is trivially algebraic-simulation-sound extractable even without any condition on the bases.

D. Algebraic Simulation-Sound Extractability of Schnorr Proofs

Weak Algebraic Simulation-Sound Extractability. Let us first define the extractor $\mathrm{Ext}^{(g_1,\ldots,g_n)}(\sigma,\tau,x,\pi,\ell)$ as follows: it first checks the proof π (using VK), and returns $\perp$ if this proof is not valid. Otherwise, it gets the discrete logarithms $\vec{\lambda_u}=(\lambda_{u,1},\ldots,\lambda_{u,n})\in\mathbb{Z}_p^n$ for u (in base $(g_1,\ldots,g_n)$) and $\vec{\lambda_h}=(\lambda_{h,1},\ldots,\lambda_{h,n})\in\mathbb{Z}_p^n$ of h, and returns r, such that $\vec{\lambda_u}=r\vec{\lambda_h}$, if possible, and $\perp$ otherwise. When the base is $(g=g_1)\xleftarrow{R}G\setminus\{1\}$, if $h\neq 1$, $\vec{\lambda_h}\in\mathbb{Z}_p$ and $\vec{\lambda_h}\neq 0$, therefore r always exists and the extractor clearly always output the correct answer. If $h=1$ and $u\neq 1$, then, with probability $1-1/p$, $H(\ell,x,u')\neq\log_u u'$, and so $\mathsf{Adv}^{\text{w-a-ext}}(\mathcal{A})\leq 1-(1-1/p)^{n_{\mathrm{ro}}}\leq n_{\mathrm{ro}}/p$, where n_{ro} is the number of random oracle queries.

Base Indistinguishability. To prove base indistinguishability, we first remark that when Ext does not return $\perp$, it will return the same value with $(g_1,\ldots,g_n)$ as with (g_1), as it always return either $\perp$ or the correct discrete logarithm of u in base h. With the base (g_1), the extraction always works (as seen previously in the proof of weak algebraic simulation-sound extractability). Therefore, we only need to prove that with any hard linear base $(g_1,\ldots,g_n)$, if $\pi=(z,e)$ is a valid proof for $x=(u,h)$ and ℓ, then the probability that $\mathrm{Ext}^{(g_1,\ldots,g_n)}(\sigma,\tau,x,\pi,\ell)$ returns $\perp$ is negligible.

For that purpose, we write $\vec{\lambda_u}$, $\vec{\lambda_{u'}}$, and $\vec{\lambda_h}$ the discrete logarithms of u, $u'=h^z u^e$, and h in base $(g_1,\ldots,g_n)$. For any u, u', h, and ℓ, if $\vec{\lambda_u}$ and $\vec{\lambda_h}$ are not linearly dependent (i.e., if we cannot extract the proof for (u,h)), there exists at most one scalar $e\in\mathbb{Z}_p$, such that $\vec{\lambda_{u'}}-e\vec{\lambda_u}$ and $\vec{\lambda_h}$ are linearly dependent. If $H(\ell,(u,h),u')\xleftarrow{R}\mathbb{Z}_p$ does not output that value e, and $\pi=(z,e)$ is valid for word $x=(u,h)$ and label ℓ, then $u'=h^z u^e$ and one can find $\vec{\mu}=\vec{\lambda_{u'}}-z\vec{\lambda_h}-e\vec{\lambda_u}\neq 0^n$ where $g_1^{\mu_1}\cdots g_n^{\mu_n}=1$. In other words, if we cannot extract the witness from a valid proof $\pi=(z,e)$ for word $x=(u,h)$ and label ℓ, then with probability $\geq(1-1/p)^{n_{\mathrm{ro}}}\geq 1-n_{\mathrm{ro}}/p$ (the probability that for any query to the random oracle, the output is not the bad value e), we can extract a vector $\vec{\mu}$ breaking the hard linear property of $\mathcal{D}$. Hence:

$$\mathsf{Adv}^{\text{base-ind}}_{\mathsf{NIZK},\mathcal{D}}(\mathcal{A})\leq\mathsf{Adv}^{\text{hard-lin}}_{\mathcal{D}}(t')+n_{\mathrm{ro}}/p$$

with $t'=t+O(n_{\mathrm{o}}t_{\mathsf{exp}})$, n_{o} the number of queries to the Sim_2 and Ext oracles, and t the running time of $\mathcal{A}$.

E. Update of the J-PAKE Security Proof

Overview. We first remark that our reduction for J-PAKE is algebraic: in each protocol of the proof in Section VI, instances are algebraic. To prove the J-PAKE security with algebraic-simulation-sound extractable proofs, it is therefore sufficient to exhibit which bases are used in each protocol (which basically correspond to the group elements coming from the assumptions used) and to show that all these bases are hard-linear. Bases just consist of the group elements coming from the hard problems used in the proof: DDH, DSDH, and DTGDH (which can be replaced by three DDH).

More precisely, when we are doing a reduction from a distinguisher between protocols P_i and P_{i+1}, to e.g., DDH, what we do is, from a tuple $(g_1=g,g_2=X,g_3=Y,g_4=Z)\in G^4$, simulate instances of the protocol such that, if $Z=\mathsf{DH}(X,Y)$ (DDH tuple) then we simulate everything as in P_i, while if Z is random (random tuple), then we simulate everything as in P_{i+1}. So we just consider a protocol P_i^+ which is as P_i except every group element is computed from a DDH tuple $(g_1,\ldots,g_4)$ as in the reduction, and a protocol P_{i+1}^- which is as P_{i+1} except every group element is computed from a random tuple $(g_1,\ldots,g_4)$. In addition, we suppose that in P_i and P_{i+1}, the extraction Ext is made in base (g_1), while in P_i^+ and P_{i+1}^-, the extraction is made in base $(g_1,\ldots,g_4)$. Then the DDH assumption ensures that P_i^+ and P_{i+1}^- are indistinguishable (as in the original proof), while the base indistinguishability from DDH tuples $(g_1,\ldots,g_4)$ and random tuples $(g_1,\ldots,g_4)$ ensures that P_i and P_i^+ are indistinguishable and P_{i+1}^- and P_{i+1} are indistinguishable, assuming (as proved below) that these bases are hard-linear. So finally: P_i and P_{i+1} are indistinguishable under base indistinguishability and DDH.

From the above discussion, we only need to prove that the following bases are hard-linear:

- $(g_1 = g, g_2 = X = g^x, g_3 = Y = g^y, g_4 = Z)$ with $g \xleftarrow{R} G \setminus \{1\}$, $x, y \xleftarrow{R} \mathbb{Z}_p$, and $Z = g^{xy}$ in the one hand, or $Z \xleftarrow{R} G$ on the other hand (for DDH);
- $(g_1 = g, g_2 = X = g^x, g_3 = Z = g^{x^2})$ with $g \xleftarrow{R} G \setminus \{1\}$, $x \xleftarrow{R} \mathbb{Z}_p$, and $Z = g^{x^2}$ on the one hand, or $Z \xleftarrow{R} G$ on the other hand (for DSDH);
- $(g_1 = g, g_2 = X = g^x, g_3 = Y = g^y, g_4 = Z = g^z, g_5 = g^{xy}, g_6 = g^{xz}, g_7 = g^{yz}, g_8 = Z)$ with $g \xleftarrow{R} G \setminus \{1\}$, $x, y, z \xleftarrow{R} \mathbb{Z}_p$, and $Z = g^{xyz}$ on the one hand, or $Z \xleftarrow{R} G$ on the other hand (for DTGDH).

Hard linearity of random bases. Let us first prove the hard linearity of random bases $(g_1 = g, g_2, \ldots, g_n) \xleftarrow{R} \mathcal{D} = (G \setminus \{1\}) \times G^{n-1}$, under the discrete logarithm assumption, which states it is hard to compute the discrete logarithm of a random group element $X \in G$ in base g, a generator of G.

The reduction first picks $\alpha_2, \beta_2, \ldots, \alpha_n, \beta_n \xleftarrow{R} \mathbb{Z}_p$, sets $g_i = g^{\alpha_i} X^{\beta_i}$ for $i = 2, \ldots, n$, and gives it to the adversary for the hard linearity problem $\mathcal{A}$. Then $\mathcal{A}$ outputs $(\mu_1, \ldots, \mu_n) \neq (0, \ldots, 0)$, such that $g_1^{\mu_1} \cdots g_n^{\mu_n} = 1$. If we take the discrete logarithm of the previous relation

$$(\mu_1 + \mu_2\alpha_2 + \cdots + \mu_n\alpha_n) + (\mu_2\beta_2 + \cdots + \mu_n\beta_n)x = 0,$$

where x is the discrete logarithm of X in base g, i.e., the answer to the discrete logarithm instance X. Since $(\mu_1, \ldots, \mu_n) \neq (0, \ldots, 0)$, there exists $i = 2, \ldots, n$ such that $\mu_i \neq 0$, and in addition, $(\beta_2, \ldots, \beta_n)$ are hidden to the adversary (from an information theoretic point of view), hence $\mu_2\beta_2 + \cdots + \mu_n\beta_n = 0$ with probability $1/p$. And when, $\mu_2\beta_2 + \cdots + \mu_n\beta_n \neq 0$, the reduction can compute $x = -(\mu_1 + \mu_2\alpha_2 + \cdots + \mu_n\alpha_n)/(\mu_2\beta_2 + \cdots + \mu_n\beta_n)$.

Hence:

$$\mathsf{Adv}^{\text{hard-lin}}_{\mathcal{D}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{DL}}_{G}(t') + 1/p,$$

with $t' = t + O(nt_{\mathsf{exp}})$, and t the running time of $\mathcal{A}$.

Hard linearity of bases used in the proof. To prove the hard linearity of the bases used in the proof, it is therefore sufficient to prove that they are computationally indistinguishable from random bases. Clearly, bases from DDH and DSDH are either random or indistinguishable from random.

It remains to show that bases $(g_1 = g, g_2 = X = g^x, g_3 = Y = g^y, g_4 = Z = g^z, g_5 = g^{xy}, g_6 = g^{xz}, g_7 = g^{yz}, g_8 = Z)$ with $g \xleftarrow{R} G \setminus \{1\}$, $x, y, z \xleftarrow{R} \mathbb{Z}_p$, and $Z = g^{xyz}$ on the one hand, or $Z \xleftarrow{R} G$ on the other hand, for DTGDH are hard linear. They are indistinguishable from $(g_1 = g, g_2 = X = g^x, g_3 = Y = g^y, g_4 = Z = g^z, g_5 = g^{x'}, g_6 = g^{y'}, g_7 = g^{z'}, g_8 = Z)$, with $g \xleftarrow{R} G \setminus \{1\}$, $x, y, z, x', y', z' \xleftarrow{R} \mathbb{Z}_p$, with $Z = g^{xyz}$ or $Z \xleftarrow{R} G$ respectively, under the DDH assumption (properly randomized). When $Z \xleftarrow{R} G$, this is a random tuple, otherwise this is indistinguishable from a random tuple under DDH, so finally, $\mathsf{Adv}^{\text{hard-lin}}_{\mathcal{D}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{DL}}_{G}(t') + 1/p + 2\mathsf{Adv}^{\mathrm{DDH}}_{G}(t')$, for $\mathcal{D}$ any of the two base distributions for DTGDH, and $t' = t + O(nt_{\mathsf{exp}})$.

Putting this all together, when the J-PAKE protocol is instantiated with Schnorr proofs of knowledge, and we work in a model with algebraic adversaries and random oracles, we can plug the following bounds into Theorem VI.1 and Theorem VII.1:

$$\mathsf{Adv}^{\mathrm{uzk}}_{\mathsf{NIZK}}(t', n_{\mathrm{sim}}) \leq (n_{\mathrm{sim}} + n_{\mathrm{ro}})/p,$$

and

$$\mathsf{Adv}^{\mathrm{ext}}_{\mathsf{NIZK}}(t', n_{\mathrm{sim}}) = \mathsf{Adv}^{\text{w-a-ext}}(t', n_{\mathrm{sim}}) \leq n_{\mathrm{ro}}/p,$$

where n_{ro} is the number of queries to the random oracle made by the adversary $\mathcal{A}$. Technically we should also add the advantage $\mathsf{Adv}^{\text{base-ind}}$ each time the base changes, but this only adds $O(\mathsf{Adv}^{\mathrm{DDH}}(t') + \mathsf{Adv}^{\mathrm{DSDH}}(t') + \mathsf{Adv}^{\mathrm{DL}}(t') + n_{\mathrm{ro}}/p) = O(\mathsf{Adv}^{\mathrm{DSDH}}(t'))$ which is a term already present in the original bound.

ACKNOWLEDGMENTS

This work was supported in part by 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

[1] M. Abdalla, F. Benhamouda, and D. Pointcheval, "Public-key encryption indistinguishable under plaintext-checkable attacks," in *PKC 2015*, ser. LNCS, J. Katz, Ed., vol. 9020. Springer, Mar. / Apr. 2015, pp. 332–352.

[2] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *PKC 2005*, ser. LNCS, S. Vaudenay, Ed., vol. 3386. Springer, Jan. 2005, pp. 65–84.

[3] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," in *CT-RSA 2005*, ser. LNCS, A. Menezes, Ed., vol. 3376. Springer, Feb. 2005, pp. 191–208.

[4] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *ICICS 03*, ser. LNCS, S. Qing, D. Gollmann, and J. Zhou, Eds., vol. 2836. Springer, Oct. 2003, pp. 301–312.

[5] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *EUROCRYPT 2000*, ser. LNCS, B. Preneel, Ed., vol. 1807. Springer, May 2000, pp. 139–155.

[6] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS 93*, V. Ashby, Ed. ACM Press, Nov. 1993, pp. 62–73.

[7] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 1992, pp. 72–84.

[8] ——, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," in *ACM CCS 93*, V. Ashby, Ed. ACM Press, Nov. 1993, pp. 244–250.

[9] D. Boneh, "The decision Diffie-Hellman problem," in *Algorithmic number theory*. LNCS, 1998, pp. 48–63.

[10] V. Boyko, P. D. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," in *EUROCRYPT 2000*, ser. LNCS, B. Preneel, Ed., vol. 1807. Springer, May 2000, pp. 156–171.

[11] E. Bresson, O. Chevassut, and D. Pointcheval, "New security results on encrypted key exchange," in *PKC 2004*, ser. LNCS, F. Bao, R. Deng, and J. Zhou, Eds., vol. 2947. Springer, Mar. 2004, pp. 145–158.

[12] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie, "Universally composable password-based key exchange," in *EUROCRYPT 2005*, ser. LNCS, R. Cramer, Ed., vol. 3494. Springer, May 2005, pp. 404–421.

[13] C. Chevalier, P.-A. Fouque, D. Pointcheval, and S. Zimmer, "Optimal randomness extraction from a Diffie-Hellman element," in *EUROCRYPT 2009*, ser. LNCS, A. Joux, Ed., vol. 5479. Springer, Apr. 2009, pp. 572–589.

[14] Y. Cliff, C. Boyd, and J. M. González Nieto, "How to extract and expand randomness: A summary and explanation of existing results," in *ACNS 09*, ser. LNCS, M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, Eds., vol. 5536. Springer, Jun. 2009, pp. 53–70.

[15] J.-S. Coron, J. Patarin, and Y. Seurin, "The random oracle model and the ideal cipher model are equivalent," in *CRYPTO 2008*, ser. LNCS, D. Wagner, Ed., vol. 5157. Springer, Aug. 2008, pp. 1–20.
[16] "Firefox Sync." [Online]. Available: https://www.mozilla.org/en-US/firefox/sync/
[17] J. A. Garay, P. D. MacKenzie, and K. Yang, "Strengthening zero-knowledge protocols using signatures," *Journal of Cryptology*, vol. 19, no. 2, pp. 169–209, Apr. 2006.
[18] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange," *ACM Transactions on Information and System Security*, vol. 9, no. 2, pp. 181–234, 2006.
[19] O. Goldreich and Y. Lindell, "Session-key generation using human passwords only," in *CRYPTO 2001*, ser. LNCS, J. Kilian, Ed., vol. 2139. Springer, Aug. 2001, pp. 408–432.
[20] V. Goyal, A. Jain, and R. Ostrovsky, "Password-authenticated session-key generation on the internet in the plain model," in *CRYPTO 2010*, ser. LNCS, T. Rabin, Ed., vol. 6223. Springer, Aug. 2010, pp. 277–294.
[21] A. Groce and J. Katz, "A new framework for efficient password-based authenticated key exchange," in *ACM CCS 10*, E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, Eds. ACM Press, Oct. 2010, pp. 516–525.
[22] J. Groth, "Simulation-sound NIZK proofs for a practical language and constant size group signatures," in *ASIACRYPT 2006*, ser. LNCS, X. Lai and K. Chen, Eds., vol. 4284. Springer, Dec. 2006, pp. 444–459.
[23] J. Groth, R. Ostrovsky, and A. Sahai, "Perfect non-interactive zero knowledge for NP," in *EUROCRYPT 2006*, ser. LNCS, S. Vaudenay, Ed., vol. 4004. Springer, May / Jun. 2006, pp. 339–358.
[24] F. Hao and P. Ryan, "J-pake: Authenticated key exchange without pki," in *Transactions on Computational Science XI*, ser. Lecture Notes in Computer Science, M. Gavrilova, C. Tan, and E. Moreno, Eds. LNCS, 2010, vol. 6480, pp. 192–206.
[25] F. Hao and P. Zielinski, "A 2-round anonymous veto protocol," in *Security Protocols, 14th International Workshop, Cambridge, UK, March 27-29, 2006, Revised Selected Papers*, ser. Lecture Notes in Computer Science, B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, Eds., vol. 5087. LNCS, 2006, pp. 202–211.
[26] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1364–1396, 1999.
[27] T. Holenstein, R. Künzler, and S. Tessaro, "The equivalence of the random oracle model and the ideal cipher model, revisited," in *43rd ACM STOC*, L. Fortnow and S. P. Vadhan, Eds. ACM Press, Jun. 2011, pp. 89–98.
[28] D. P. Jablon. [Online]. Available: http://www.jablon.org/passwordlinks.html
[29] ——, "Strong password-only authenticated key exchange," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 5, pp. 5–26, Oct. 1996.
[30] S. Jiang and G. Gong, "Password based key exchange with mutual authentication," in *SAC 2004*, ser. LNCS, H. Handschuh and A. Hasan, Eds., vol. 3357. Springer, Aug. 2004, pp. 267–279.
[31] J. Katz. [Online]. Available: https://www.lightbluetouchpaper.org/2008/05/29/j-pake/\#comment-9547
[32] J. Katz, R. Ostrovsky, and M. Yung, "Efficient and secure authenticated key exchange using weak passwords," *Journal of the ACM*, vol. 57, no. 1, 2009.
[33] J. Katz and V. Vaikuntanathan, "Round-optimal password-based authenticated key exchange," in *TCC 2011*, ser. LNCS, Y. Ishai, Ed., vol. 6597. Springer, Mar. 2011, pp. 293–310.
[34] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme," in *CRYPTO 2010*, ser. LNCS, T. Rabin, Ed., vol. 6223. Springer, Aug. 2010, pp. 631–648.
[35] P. MacKenzie, "On the security of the SPEKE password-authenticated key exchange protocol," Cryptology ePrint Archive, Report 2001/057, 2001. [Online]. Available: http://eprint.iacr.org/2001/057
[36] P. D. MacKenzie, S. Patel, and R. Swaminathan, "Password-authenticated key exchange based on RSA," in *ASIACRYPT 2000*, ser. LNCS, T. Okamoto, Ed., vol. 1976. Springer, Dec. 2000, pp. 599–613.
[37] P. D. MacKenzie and K. Yang, "On simulation-sound trapdoor commitments," in *EUROCRYPT 2004*, ser. LNCS, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, May 2004, pp. 382–400.
[38] "Nest." [Online]. Available: http://nest.com
[39] "OpenSSL project." [Online]. Available: http://www.openssl.org
[40] P. Paillier and D. Vergnaud, "Discrete-log-based signatures may not be equivalent to discrete log," in *ASIACRYPT 2005*, ser. LNCS, B. K. Roy, Ed., vol. 3788. Springer, Dec. 2005, pp. 1–20.
[41] S. Patel, "Number theoretic attacks on secure password schemes," in *1997 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1997, pp. 236–247.
[42] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.
[43] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.
[44] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *EUROCRYPT'97*, ser. LNCS, W. Fumy, Ed., vol. 1233. Springer, May 1997, pp. 256–266.
[45] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," in *ACM CCS 96*. ACM Press, Mar. 1996, pp. 31–37.
[46] "Thread protocol." [Online]. Available: http://www.threadgroup.org
[47] T. D. Wu, "The secure remote password protocol," in *NDSS'98*. The Internet Society, Mar. 1998.

APPENDIX
THE INVERTED-ADDITIVE DIFFIE-HELLMAN ASSUMPTIONS (CIDH AND DIDH)

Here we recall two assumptions used in [35] to prove the security of the SPEKE protocols (for the comparison in Table I), and show some relations between these assumptions and the other assumptions we consider in this paper. These relations are summarized in Fig. 2.

A. Definition of CIDH and DIDH

Computational Inverted-Additive Diffie-Hellman (CIDH). For two values $X = g^x$ and $Y = g^y$, let $\mathsf{IDH}(X, Y) = g^{xy/(x+y)}$ (or 1 if $x + y = 0$) be the Inverted Additive Diffie-Hellman value corresponding to X and Y. An algorithm for the Computational Inverted-Additive Diffie-Hellman takes two elements X and Y, and outputs the Inverted-Additive Diffie-Hellman value of X and Y. Let $\mathcal{A}$ be an algorithm with input (X, Y). Let $\mathsf{Adv}_G^{\mathrm{CIDH}}(\mathcal{A})$ be

$$\Pr\left[(x,y) \xleftarrow{R} \mathbb{Z}_p^2;\, X \leftarrow g^x;\, Y \leftarrow g^y : \mathcal{A}(X,Y) = \mathsf{IDH}(X,Y)\right].$$

Decision Inverted-Additive Diffie-Hellman (DIDH). An algorithm for the Decision Inverted-Additive Diffie-Hellman takes three elements X, Y, and Z, and attempts to distinguish whether Z is the Inverted-Additive Diffie-Hellman value corresponding to X and Y, or is a random element of G. Let $\mathcal{A}$ be an algorithm with input (X, Y, Z). Let $\mathsf{Adv}_G^{\mathrm{DIDH}}(\mathcal{A})$ be

$$\Pr\left[(x,y) \xleftarrow{R} \mathbb{Z}_p^2;\, X \leftarrow g^x;\, Y \leftarrow g^y;\, Z \leftarrow \mathsf{IDH}(X,Y) : \mathcal{A}(X,Y,Z) = 1\right] - \Pr\left[(x,y,z) \xleftarrow{R} \mathbb{Z}_p^3;\, X \leftarrow g^x;\, Y \leftarrow g^y;\, Z \leftarrow g^z : \mathcal{A}(X,Y,Z) = 1\right].$$

B. Relations

Hardness of DSDH Implies Hardness of DIDH. Let $\mathcal{A}$ be an adversary against DIDH. Then we can construct an adversary $\mathcal{B}$ against DSDH as follows: $\mathcal{B}(X, Y)$ picks

$z \xleftarrow{R} \mathbb{Z}_p$ and outputs $\mathcal{A}(X', Y', Z')$ with $(X', Y', Z') = (g^z X, g^z X^{-1}, (g^{z^2} Y^{-1})^{1/(2z)})$ ($Z' = 0$ when $z = 0$).

Indeed, if $x \xleftarrow{R} \mathbb{Z}_p$, $X = g^x$, and $Y = \mathsf{SDH}(X) = g^{x^2}$, then $(X', Y', Z') = (g^{x'}, g^{y'}, g^{z'})$, with $x' = z+x$, $y' = z-x$, $z' = (z^2 - x^2)/(2z) = x'y'/(x'+y')$. Therefore, $Z' = \mathsf{IDH}(X, Y)$, and X' and Y' are two independent uniform random group elements in G, as $(x, z) \mapsto (z + x, z - x)$ is a bijection of $\mathbb{Z}_p^2$.

Otherwise, if $x, y \xleftarrow{R} \mathbb{Z}_p$, $(X, Y) = (g^x, g^y)$, then $(X', Y', Z') = (g^{x'}, g^{y'}, g^{z'})$, with $x' = z + x$, $y' = z - x$, $z' = (z^2 - y)/(2z)$. The function $(x, y, z) \mapsto (z + x, z - x, (z^2 - y)/(2z))$ is a bijection of $\mathbb{Z}_p^3$, if we exceptionally set $(z^2 - y)/(2z) = y$ when $z = 0$. Therefore, (X, Y, Z) are $1/p$-statistically close to the uniform distribution of G^3.

Finally, we get:

$$\mathsf{Adv}_G^{\mathrm{DIDH}}(\mathcal{A}) \leq \mathsf{Adv}_G^{\mathrm{DSDH}}(\mathcal{B}) + 1/p.$$

Hardness of DIDH Implies Hardness of CIDH. It is clear that we have $\mathsf{Adv}_G^{\mathrm{DIDH}}(t) \leq \mathsf{Adv}_G^{\mathrm{CIDH}}(t) + 1/p$.

Hardness of CSDH Implies Hardness of CIDH. The proof is very similar to the one that DSDH implies DIDH. More precisely, let $\mathcal{A}$ be an adversary against CIDH. Then we can construct an adversary $\mathcal{B}$ against CSDH as follows: $\mathcal{B}(X)$ picks $z \xleftarrow{R} \mathbb{Z}_p$, computes $Z' \xleftarrow{R} \mathcal{A}(X', Y')$ with $(X', Y') = (g^z X, g^z X^{-1})$, and outputs $Y = {Z'}^{-2z} g^{z^2}$.

With probability $\mathsf{Adv}_G^{\mathrm{CIDH}}(\mathcal{A})$, we have $Z' = \mathsf{IDH}(X', Y') = (z^2 - x^2)/(2z)$, hence $Y = g^{x^2}$, where $X = g^x$, except if $z = 0$. Therefore, we get

$$\mathsf{Adv}_G^{\mathrm{CIDH}}(\mathcal{A}) \leq \mathsf{Adv}_G^{\mathrm{CSDH}}(\mathcal{B}) + 1/p.$$

Hardness of CIDH Implies Hardness of DL. It is clear that we have $\left(\mathsf{Adv}_G^{\mathrm{DL}}(t)\right)^2 \leq \mathsf{Adv}_G^{\mathrm{CIDH}}(t)$, as knowing the discrete logarithms of X and Y enables to compute $\mathsf{IDH}(X, Y)$.