Universally Composable Password-Based Key Exchange

Ran Canetti[1⋆], Shai Halevi[1], Jonathan Katz[2⋆⋆], Yehuda Lindell[3⋆⋆⋆], and Phil MacKenzie[4†]

[1] IBM T.J. Watson Research Center, Hawthorne, NY, USA.
canetti@watson.ibm.com, shaih@alum.mit.edu
[2] Dept. of Computer Science, University of Maryland, MD, USA.
jkatz@cs.umd.edu
[3] Department of Computer Science, Bar-Ilan University, Israel.
lindell@cs.biu.ac.il
[4] Bell Labs, Lucent Technologies, Murray Hill, NJ, USA.

Abstract. We propose and realize a definition of security for password-based key exchange within the framework of universally composable (UC) security, thus providing security guarantees under arbitrary composition with other protocols. In addition, our definition captures some aspects of the problem that were not adequately addressed by most prior notions. For instance, it does not assume any underlying probability distribution on passwords, nor does it assume independence between passwords chosen by different parties. We also formulate a definition of password-based secure channels, and show that such a definition is achievable given password-based key exchange.

Our protocol realizing the new definition of password-based key exchange is in the common reference string model and relies on standard number-theoretic assumptions. The components of our protocol can be instantiated to give a relatively efficient solution which is conceivably usable in practice. We also show that it is impossible to satisfy our definition in the "plain" model (e.g., without a common reference string).

1 Introduction

Protocols for password-based key exchange have received much attention in recent years. In short, the problem is how to enable authenticated generation of a "high-quality" secret key between two parties whose only *a priori* shared, secret information consists of a low-entropy password. In this setting, an attacker can always correctly determine the correct password via an *on-line dictionary attack* in which the adversary exhaustively enumerates the password space and tries to impersonate one of the parties using each possible shared secret. Since such an attack is unavoidable, work in this area focuses on preventing *off-line dictionary attacks*. Roughly, this guarantees that the exhaustive, on-line attack is the

⋆ Supported by NSF CyberTrust Grant 0430450.
⋆⋆ Supported by NSF CAREER award 0447075 and Trusted Computing grant 0310751.
⋆⋆⋆ Some of this work was carried out while the author was at IBM T.J. Watson.
† Current addr.: DoCoMo USA Labs, San Jose, CA. (philmac@docomolabs-usa.com)

"best" possible one. That is, the attacker must interact with a legitimate player in order to verify each password guess, and the interaction leaks no information other than whether or not the attacker's guess is correct. Besides their practical importance, password-based protocols are also interesting from a purely theoretical point of view: they provide a rare case where bootstrapping "strong security" from "weak security" has to be modeled, obtained, and argued.

The problem of resistance to off-line password-guessing attacks was first raised by Gong, et al. [21] in the asymmetric "PKI model" (where, in addition to a password, the user has the public key of the server). Formal definitions and proofs of security in this setting were later given by Halevi and Krawczyk [22]. A more difficult setting for this problem is one where the parties share *only* a password (and in particular, neither party knows the other's public-key). This setting was first considered by Bellovin and Merritt [5], and their work was followed by much additional research developing protocols with heuristic justifications for their security (see [6] for a survey). Formal definitions for this setting, together with protocols analyzed in the random-oracle/ideal-cipher models, were given by Bellare, et al. [3] (who proposed an indistinguishability-based definition) and Boyko, et al. [7] (who proposed a simulation-based definition). Goldreich and Lindell [19] introduced a third security definition and also gave the first provably-secure solution to this problem in the standard model, based on general assumptions; their protocol was recently simplified (at the expense of achieving a weaker security guarantee) by Nguyen and Vadhan [26]. Another setting that has been considered for this problem is one where, in addition to shared low-entropy passwords, all parties share a common reference string. In this setting, a practical and provably-secure protocol was first developed by Katz, et al. [24] based on the decisional Diffie-Hellman assumption. This protocol was subsequently generalized and abstracted by Gennaro and Lindell [18] who, among other things, obtain protocols that rely on the quadratic residuosity and N^{th}-residuosity assumptions.

The many definitions that have already been introduced [3, 7, 19, 26] indicate that finding a "good" definition of security for password-based authentication has been difficult and remains a challenging problem. Furthermore, it is not clear that any of the above definitions adequately address all aspects of the problem. For example, none of the above definitions relate to the (realistic) setting where the password-based protocol is used as a component within a larger protocol. (Rather, it is assumed that the entire network activity consists of many executions of the password protocol.) Since the problem at hand involves *non-negligible* probabilities of "success" by the adversary, providing security-preserving composition (with reasonable error propagation) is even more delicate than usual. Some of the above definitions have not been proven sufficient for implementing (any form of) secure channels — a natural goal of key-exchange protocols (see [9] for motivation). Finally, existing (explicit) definitions assume that passwords are chosen from some *pre-determined, known distribution*, and (with the exception of [7]) assume also that passwords shared between different parties are *independent*. (However, it is claimed in [24] that their proof extends to the case of dependent passwords.) These assumptions rarely hold in practice.

A new definition. In this work, we propose and realize a new definition of security for password-based key-exchange protocols within the universally composable (UC) security framework [8]. That is, we propose an ideal functionality for "password-based key exchange" that captures the security requirements of the problem. (Such an ideal functionality can be thought of as the code for a "centralized trusted service", were one actually available to the parties.) Working in the UC framework allows us to benefit from the universal composition theorem. Loosely speaking, the theorem states that a protocol secure in this framework remains secure even when run in an arbitrary network, where many different protocols (secure or not) may run concurrently. In addition to addressing composability, the definition in this work also addresses the other concerns mentioned above. In particular, security is preserved even in the case of arbitrary and unknown password distributions, and even if related passwords are used. The important feature here is that the probability of the adversary succeeding in its attack is negligibly close to the probability of its guessing the password outright, even when this guess is based on information about the password that the adversary obtains from the arbitrary network or from related passwords that it has (either partially or completely) learned. Finally, we show how protocols satisfying our definition may be used to construct (password-based) secure channels. Such channels enable private and authenticated communication, which in most cases is the goal of running the protocol in the first place.

As one might expect, formulating an ideal functionality that captures all the requirements of password-based key exchange involves a number of non-trivial definitional choices. Our formulation builds on the known UC formulation of (standard) key-exchange [8, 9], where security is guaranteed except with negligible probability. However, unlike standard key-exchange, some mechanism must be introduced that allows the adversary to "break" the protocols with some non-negligible probability by guessing the correct password. A natural way of doing this is to have the functionality choose the passwords for the parties. Then, if the adversary correctly guesses the password (where this guess is given to the functionality), it is allowed to choose the session-key that the parties obtain. Although this formulation is quite intuitive, it is somewhat limited in that it assumes some pre-determined dictionary or distribution on passwords and that passwords are chosen independently from each other. It also fails to model possible leakage of a portion of the password to the adversary (this is due to the fact that only the functionality knows the password).

We therefore take a different approach and allow the calling protocol (or the environment) to provide the password as part of the input. While this formulation may seem somewhat counter-intuitive at first, we show that it results in a definition of security that is at least as strong as that given by the first formulation.[1] Furthermore, it does not make any assumptions as to how the password is chosen and it imposes no pre-determined probabilities of failure.

[1] The alternative formulation in which the functionality chooses the passwords was omitted for lack of space. It may be obtained from the authors, along with a proof that it is implied by the definition presented here.

Realizing the definition. We construct a protocol that realizes our definition. The protocol is an extension of the protocols of [24, 18], and as such is in the common reference string model and may be based on some standard number-theoretic assumptions (namely the decisional Diffie-Hellman, quadratic residuosity, or N^{th}-residuosity assumptions). Our protocol uses building blocks that have efficient instantiations under these assumptions. As a result, our protocol is reasonably efficient and is realizable in practice (it has 6 rounds and at most 30 modular exponentiations per party). Some of the efficiency improvements we use in our protocol are applicable also to the protocol of [24] (and seemingly [18]); see [23]. Applied there, these improvements yield the most efficient known password-based protocol meeting the definition of [3] without random oracles.

On the necessity of set-up assumptions. Our protocol is constructed in the common reference string model, and so requires a trusted setup phase. In fact, we show that our UC-based definition of password-based key-exchange cannot be securely realized by any protocol in the *plain model* (i.e., in a model with no trusted setup whatsoever). Beyond providing some justification for our use of a common reference string, this result stands in sharp contrast with the fact that standard UC-secure key exchange *can* be realized in the plain model [9]. It also shows that our definition is strictly stronger than the definition used by [19, 26], which *can* be realized in the plain model. (We stress that in contrast to our definition, the definitions of [19, 26] do not guarantee security even under concurrent composition of the same protocol with the same password.)

Password-based secure channels. Perhaps the most important application of key-exchange protocols is for establishing secure communication sessions between pairs of parties. To advocate the adequacy of our proposed definition we formulate a UC notion of password-based secure channels, and show how to realize it given our notion of password-based key exchange. It is of course impossible to obtain standard secure channels using short passwords, since the adversary may guess the password with non-negligible probability. Consequently, our notion of password-based secure channels relaxes the standard notion in a way similar to which our notion of password-based key exchange relaxes the standard notion of key exchange. We then show that the standard protocols for realizing secure channels based on standard key exchange (see, e.g., [9]), suffice also for realizing password-based secure channels from password-based key exchange.

Organization. Due to lack of space in this abstract, the main text focuses on our definition and a high-level description of our protocol, along with motivation as to its security. The full description of the protocol and its proof of security is provided in the full version. Other results (namely, the impossibility of realizing our definition in the plain model and the fact that secure channels are implied) are described briefly in Section 4 and can be found in the full version.

2 Definitions of Security

In this section, we motivate and present our formulation of an ideal functionality for password-based key exchange in the UC framework. We stress that from here on, when we say that a protocol securely realizes some functionality, we mean that it securely realizes it according to the definitions of the UC framework. Our presentation assumes familiarity the UC framework; see [8] for a full description.

2.1 High-Level Approach

The starting point for our approach is the definition for universally composable "standard" key-exchange [9] (cf. Figure 1). Our aim is to define a functionality that achieves the same effect as standard key-exchange (where the parties have high-entropy keys), except that we also incorporate the inherent "security defect" due to the use of low-entropy passwords. Two ways of introducing this "security defect" come to mind:

1. One option is to consider the same functionality $\mathcal{F}_{\mathsf{KE}}$ as in Figure 1, but to relax the requirement of indistinguishability between the real and ideal worlds. I.e., when passwords are assumed to be chosen uniformly from a dictionary $\mathcal{D}$, one would define a secure protocol as one whose real-world execution is distinguishable from an interaction with the ideal functionality with probability at most, say, $1/|\mathcal{D}|$ plus a negligible amount.
2. A second possibility is to incorporate the "defect" directly into the functionality, e.g., by allowing the adversary to make explicit password guesses and to "break" the protocol following a successful guess. Here, the adversary "breaks" the protocol with noticeable probability even in the ideal world, and thus the standard notion of realizing an ideal functionality can be used.

Among previous works that used simulation-based definitions of security for password protocols, the first approach was taken by [19, 26], while the second was taken by [7]. (Other definitions are not simulation-based and so do not fit into either approach.) In this work we adopt the second option, for two reasons. First, this allows us to use the UC composition theorem and thus guarantee security of password-based key-exchange protocols even when run in arbitrary protocol environments. Second, this approach easily extends to handle additional complexities such as multiple users with different distributions on their passwords, or dependencies among various passwords. These aspects seem hard (if not impossible) to handle using the first approach.

Before proceeding to our definition, we describe the standard key-exchange functionality of [9]. (We note that the formulation of $\mathcal{F}_{\mathsf{KE}}$ in Figure 1 is somewhat different from the one in [9]; however, the differences are inconsequential for the purpose of this work.) The main idea behind the $\mathcal{F}_{\mathsf{KE}}$ functionality is as follows: If both participating parties are not corrupted, then they receive the same uniformly distributed session-key, and the adversary learns nothing of the key except that it was generated. However, if one of the parties *is* corrupted, then the adversary is given the power to fully determine the session-key. The rationale for this is that the aim of key-exchange is to enable honest parties to generate a key that is *unknown* to an external adversary. If one of the participating parties is corrupted, then the adversary will learn the generated key (because it is one of the participants), and so the security requirement is meaningless. In such a case, there is nothing lost by allowing the adversary to determine the key. We remark that the "role" variable in the NewSession message is included in order to let a party know if it is playing the initiator or responder role in the protocol. This has no effect on the security, but is needed for correct executions.

Functionality $\mathcal{F}_{\mathsf{KE}}$

$\mathcal{F}_{\mathsf{KE}}$ is parameterized by a security parameter k. It interacts with an adversary S and a set of (dummy) parties via the following queries.

Upon receiving a query $(\mathsf{NewSession}, sid, P_i, P_j, \text{role})$ **from party** P_i**:**
Send $(\mathsf{NewSession}, sid, P_i, P_j, \text{role})$ to S. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record (P_j, P_i), then record (P_i, P_j).

Upon receiving a query $(\mathsf{NewKey}, sid, P_i, sk)$ **from** S**, where** $|sk| = k$**:**
If there is a record (P_i, P_j), and this is the first NewKey query for P_i, then:
- If either P_i or P_j is corrupted, then output (sid, sk) to player P_i.
- If there is also a record (P_j, P_i), and a key sk' was sent to P_j, output (sid, sk') to P_i.
- In any other case, pick a new random key sk' of length k and send (sid, sk') to P_i.

Fig. 1. The authenticated key-exchange functionality $\mathcal{F}_{\mathsf{KE}}$

We now proceed to our definition of the password-based key-exchange functionality $\mathcal{F}_{\mathsf{pwKE}}$. Similarly to $\mathcal{F}_{\mathsf{KE}}$, if one of the participating parties is corrupted the adversary is given the power to fully determine the resulting session-key. However, this power is also given to the adversary in case it succeeds in guessing the parties' shared password. An additional property of our definition is that *failed* adversarial attempts at guessing a key are detected by the participating parties. Specifically, if the adversary makes a wrong password guess in a given session, then the session is marked interrupted and the parties are provided independently-chosen session-keys. (Giving the parties error messages in this case would correspond to requiring explicit mutual authentication; see additional discussion below.)

In the functionality, a session is marked compromised if the adversary makes a successful password guess (as discussed above, in this case the adversary is allowed to determine the session-key). If a session is marked fresh, this means that it is neither interrupted nor compromised. Such sessions (between honest parties) conclude with both parties receiving the same, uniformly distributed session-key. See Figure 2 for the full definition of the functionality.

In the definition of $\mathcal{F}_{\mathsf{pwKE}}$, the password is chosen by the environment who then hands it to the parties as input.[2] Since we quantify over all (polynomial-time) environments, this implies that security is preserved for all efficient pass-

[2] This is in contrast to an alternative approach described in the Introduction where the functionality chooses the password according to some predetermined distribution, and this password is hidden even from the environment. As we have mentioned, security under our definition implies security under that alternative approach.

word distributions, as well as when arbitrarily related passwords are used in different session. Furthermore, since the passwords are provided by the "environment in which the protocol is run", security is preserved even when passwords are used for other, unintended purposes by that same environment. (When we say that security is preserved here, we mean that the probability that an adversary can break the password-based key-exchange protocol is the same as its probability of guessing the password outright, given the potential misuse mentioned above.) We also remark that our definition guarantees security even in the case where two honest players execute the protocol with different passwords. (In fact, this is quite a realistic scenario which occurs every time a user mistypes a password; previous definitions did not guarantee anything in such a case.)

Functionality $\mathcal{F}_{\mathsf{pwKE}}$

The functionality $\mathcal{F}_{\mathsf{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary S and a set of parties via the following queries.

Upon receiving a query $(\mathsf{NewSession}, sid, P_i, P_j, pw, \mathrm{role})$ **from party** P_i**:**
Send $(\mathsf{NewSession}, sid, P_i, P_j, \mathrm{role})$ to S. In addition, if this is the first $\mathsf{NewSession}$ query, or if this is the second $\mathsf{NewSession}$ query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark this record fresh.

Upon receiving a query $(\mathsf{TestPwd}, sid, P_i, pw')$ **from the adversary** S**:**
If there is a record of the form (P_i, P_j, pw) which is fresh, then do: If $pw = pw'$, then mark the record $\mathsf{compromised}$ and reply to S with "correct guess". If $pw \neq pw'$, then mark the record $\mathsf{interrupted}$ and reply with "wrong guess".

Upon receiving a query $(\mathsf{NewKey}, sid, P_i, sk)$ **from** S**, where** $|sk| = k$**:**
If there is a record of the form (P_i, P_j, pw), and this is the first NewKey query for P_i, then:

- If this record is $\mathsf{compromised}$, or either P_i or P_j is corrupted, then output (sid, sk) to player P_i.
- If this record is fresh, and there is a record (P_j, P_i, pw') with $pw' = pw$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was fresh at the time, then output (sid, sk') to P_i.
- In any other case, pick a new random key sk' of length k and send (sid, sk') to P_i.

Fig. 2. The password-based key-exchange functionality, $\mathcal{F}_{\mathsf{pwKE}}$

As additional justification of our definition, we show that it implies password-based secure channels (arguably the most common application of such protocols). In addition, we show that our definition implies the "expected" notion of security against a passive eavesdropper (even one who happens to know the password being used). Finally, we show that a protocol that securely realizes our functionality is secure also with respect to the definition of Bellare, et al. [3] (modulo unimportant differences regarding the formalization of session identifiers).[3] These last two results can be seen as "sanity checks" of our definition.

[3] In our formalization, a unique session identifier sid is assumed to be part of the input to the functionality. In the two-party setting, such a session identifier can be obtained by having the parties exchange random strings and then set sid to be the concatenation of these strings.

Additional discussion. The definition of $\mathcal{F}_{\mathsf{pwKE}}$ could be *strengthened* to require explicit mutual authentication by insisting that after a "wrong guess" of the password, the session would fail (instead of producing a random and independent key). Similarly, a session with mismatching passwords would also fail. We chose not to include these requirements because **(a)** we want to keep the exposition simple; **(b)** mutual authentication is not needed for secure channels; and **(c)** it is well known that any secure key-exchange protocol (including ours) can be augmented to provide mutual authentication by adding two "key confirmation" flows at the end (and refreshing the session key). The definition could also be *weakened* by notifying the simulator whether or not the passwords match in the two NewSession queries. Roughly, the difference is that the current formulation requires that an eavesdropper be unable to detect whether the session succeeded (i.e., both parties got the same key) or failed (i.e., they got different keys). Although we are not aware of any application where this is needed, it makes the definition simpler to describe and our protocol anyway satisfies this requirement.

3 A Protocol Securely Realizing $\mathcal{F}_{\mathsf{pwKE}}$

In this section, we present our protocol for securely realizing the functionality $\mathcal{F}_{\mathsf{pwKE}}$, in the common reference string model.[4] Due to lack of space in this extended abstract, many details of the protocol and its proof of security are omitted. We remark that our protocol is proven secure in the model of *static corruptions* (where the adversary may corrupt some of the participants, but only prior to the beginning of the protocol executions) and *unauthenticated channels* (where the adversary has full control over the communication channels and can insert, modify and delete all messages sent by the honest parties). We also note that although we consider static corruptions, the "weak-corruption model" of [3] is implied by our definition (and achieved by our protocol); see the full version. In the weak-corruption model, the adversary may obtain passwords adaptively throughout the execution. This essentially models leakage of passwords, rather than adaptive corruption of parties.

3.1 Preliminaries

The protocol uses a number of primitives: one-time signatures, CPA-secure and CCA-secure public-key encryption, simulation-sound zero-knowledge proofs, and smooth projective hashing. We provide only a brief description of the last two primitives here.

Simulation-sound zero-knowledge (SSZK) proofs. Informally speaking, a zero-knowledge proof system is said to be (unbounded) *simulation-sound* if it has the property that an adversary cannot provide a convincing proof for a false

[4] Our protocol actually realizes the multi-session extension $\hat{\mathcal{F}}_{\mathsf{pwKE}}$ of this functionality (see [11]). This is important for ensuring that the same common reference string can be used in all executions; see the full version for more details. For the sake of clarity, in this extended abstract we refer only to the original functionality $\mathcal{F}_{\mathsf{pwKE}}$.

statement, even if it has seen *simulated proofs*. Such simulated proofs may actually prove false statements, and so the adversary can copy these proofs but do nothing more. More formally, the adversary is given oracle access to the zero-knowledge simulator and can request simulated proofs of any statement (true or false) that it wishes. The adversary is then said to *succeed* if it generates a convincing proof of a false statement, and this proof was not received from the oracle. This concept was first introduced by Sahai [28] and De Santis, et al. [14] in the context of non-interactive zero-knowledge. For the case of interactive protocols, the notion was formally defined by Garay, et al. [17].[5] Efficient methods for transforming three-round honest-verifier zero-knowledge protocols (also called Σ-protocols [12]) into simulation-sound zero-knowledge protocols in the common reference string model have been shown in [17] and [25]. We note that, according to the definition of [17], simulation-sound zero knowledge protocols also achieve *concurrent* zero knowledge; i.e., the zero knowledge property holds for an unbounded number of asynchronous executions of an honest prover. Finally, we note that simulation-sound zero knowledge is a weaker requirement than universally-composable zero-knowledge, and more efficient constructions for it are known.

Smooth projective hashing [13]. On a very high level, a projective hash family is a family of hash functions that can be computed using one of two keys: the (secret) hashing key can be used to compute the function on every point in its domain, whereas the (public) projected key can only be used to compute the function on a specified subset of the domain. Such a family is called "smooth" if the value of the function on a value outside of the specified subset is uniformly distributed, even given the projected key. More formally (but still far from being exact), let X be a set and let $L \subset X$. We say that a hash function H_{hk} that maps X to some set is projective if there exists a projection function $\alpha(\cdot)$ that maps hash keys hk into their projections $hp = \alpha(hk)$, such that for every $x \in L$ it holds that the value of $H_{hk}(x)$ is uniquely determined by hp and x. (In contrast, for $x \notin L$ the value of $H_{hk}(x)$ need not be determined from hp and x.) A smooth projective hash function has the additional property that for $x \notin L$, hp actually says *nothing* about the value of $H_{hk}(x)$. More specifically, given x and $hp = \alpha(hk)$, the value $H_{hk}(x)$ is (statistically close to) a uniformly distributed element in the range of H_{hk}.

We already mentioned that for $x \in L$ the projected key hp fully defines the value $H_{hk}(x)$, but so far we said nothing about whether or not this value can be efficiently computed. An important property of smooth projective hash functions is that if the subset L is *an NP-language*, then for $x \in L$ it is possible to compute $H_{hk}(x)$ using the projected key $hp = \alpha(hk)$ and a witness of the fact that $x \in L$. Thus, for $x \in L$ there are two alternative ways of computing $H_{hk}(x)$:

1. Given the hashing key hk, compute $H_{hk}(x)$ directly.
2. Given the projected key hp and a witness w for $x \in L$, compute the hash value $h_{hp}(x; w) = H_{hk}(x)$.

[5] When we say a proof is simulation sound, we will also mean that it is *uniquely applicable* [28]; that is, a proof is valid for at most one statement.

Following [18], the set X that we consider in this work is the set $\{(c, m)\}$ of all ciphertext/plaintext pairs under a given public-key pke. Furthermore, the language L is taken to be $\{(c, m) \mid c = E_{pke}(m)\}$; that is, L is the set of all ciphertext/plaintext pairs (c, m) where c is an encryption of m under the public-key pke. We note this language is indeed an NP language, with the witness being the randomness that was used in the encryption of m.

We also comment that the semantic security of the encryption implies that L is hard on the average (i.e., it is hard to distinguish a random element in X from a random element in L). For such languages, it was proven in [18] that given a random $x \in L$ and $hp = \alpha(hk)$, the value $H_{hk}(x)$ is *computationally indistinguishable* from a random value in the range of H_{hk}. (This holds even though for any $x \in L$, the value $H_{hk}(x)$ is uniquely determined by x and hp.)

In the description below we denote choosing a hashing key from the family by $hk \leftarrow \mathcal{H}$, and denote the projection of this key by $hp = \alpha(hk)$. We also denote computing the hash value using the hashing key hk by $H_{hk}(x)$, and computing the hash value using the projected key hp and witness w by $h_{hp}(x; w)$. (Note that the statements x below are actually pairs (c, m), and the witness is the randomness r, so we write $H_{hk}(c, m)$ and $h_{hp}(c, m; r)$.)

3.2 The KOY/GL Protocol

The starting point of our protocol is the password-based key-exchange protocol of Katz, Ostrovsky, and Yung [24], as generalized and abstracted by Gennaro and Lindell [18]. The "core" of this protocol is sketched in Figure 3 (in this figure we suppress various details). At a high level, the parties in the KOY/GL protocol exchange CCA-secure encryptions[6] of the password, encrypted with the public-key found in the common reference string, and then compute the session key by combining (smooth projective) hashes of the two ciphertext/password pairs. In order to do this, each party chooses a hashing key for a smooth projective hash function and sends the "projected key" to the other party.

Ignoring for the moment the signature keys from Figure 3, let c_2, hk and c_1, hk' be the encryptions and hashing keys generated by parties P_i and P_j, respectively. Party P_i can compute $H_{hk}(c_1, pw)$ since it knows the actual hashing key hk. Furthermore, since it generated the ciphertext c_2, it can compute $h_{hp'}(c_2, pw; r_2) = H_{hk'}(c_2, pw)$ using its knowledge of the randomness r_2 that was used in generating $c_2 = E_{pke}(pw; r_2)$. (This relies on the two alternative ways of computing $H_{hk}(x)$ described above.) Symmetrically, P_j computes the same session key using hk', hp, and its knowledge of r_1.

The basic idea behind the security of the protocol can be described as follows. Denote the shared password of a client and server by pw. If the client receives an encryption c of the wrong password pw', then (by the definition of smooth projective hashing) the hash she computes will be random and independent of all her communication. (This holds because the statement (c, pw) is not in the

[6] It is shown in [18] that non-malleable commitments can be used in place of CCA-secure encryption. However, for our extension of the protocol to the UC framework we will need to use encryption, and so we describe it in this way.

P_i (server)		P_j (client)
	CRS: public key pke	
		$(sk, vk) \leftarrow \mathrm{sigKey}(\$)$
	$\xleftarrow{c_1,\, vk}$	$c_1 \leftarrow E_{pke}(pw; r_1)$
$c_2 \leftarrow E_{pke}(pw; r_2)$		
$hk \leftarrow \mathcal{H}$		
$hp \leftarrow \alpha(hk)$	$\xrightarrow{c_2,\, hp}$	
		$hk' \leftarrow \mathcal{H}$
		$hp' \leftarrow \alpha(hk')$
	$\xleftarrow{hp',\, \sigma}$	$\sigma \leftarrow \mathsf{Sign}_{sk}(c_2, hp, hp')$
if $(\mathsf{Verify}_{vk}((c_2, hp, hp'), \sigma) = 1)$		
session-key $\leftarrow H_{hk}(c_1, pw) + h_{hp'}(c_2, pw; r_2)$		session-key $\leftarrow h_{hp}(c_1, pw; r_1) + H_{hk'}(c_2, pw)$

Fig. 3. The core of the KOY/GL protocol.

language, so $H_{hk}(c, pw)$ is close to uniform even given the projected key hp.) A similar argument holds for the server. Thus, for an adversary to distinguish a session key from random, it must send one of the parties an encryption of the *correct* password pw.

The adversary can get an encryption of the right password by copying a ciphertext from another execution of the protocol, but then it does not know the randomness that was used to generate this ciphertext. By the property that we discussed above (regarding hard-on-the-average languages), the value $H_{hk}(c, pw)$ is computationally indistinguishable from uniform, even given hp. Moreover, since the encryption scheme is CCA-secure, and thus non-malleable, the adversary cannot generate a new encryption of pw with probability any better than it would achieve by simply guessing passwords from the dictionary and encrypting them.

Finally, the adversary may try to gain information by copying ciphertexts from a current session faithfully but not copying other values (such as the hash projected keys). This type of man-in-the-middle attack is prevented using the one-time signature. We conclude that the adversary succeeds in its attack if and only if it generates an encryption of the correct password. In other words, the adversary succeeds if it guesses the secret password, as required.

3.3 Extending the Protocol to Realize $\mathcal{F}_{\mathsf{pwKE}}$

The protocol of Figure 3 serves as a good starting point, but it does not seem to achieve the security that we require. The main issue that arises here is that an ideal-model simulator must be able to extract the adversary's password guess.[7]

[7] This need to extract is not a mere technicality, but is rather quite central to our definition. In particular, this enables us to argue that the level of security achieved is equivalent to the

At first glance, it may seem that this is not a problem because in the ideal model the simulator has control over the common reference string and so can include a public-key pke for which it knows the corresponding secret key ske. Then, when the adversary generates an encryption of the password $c = E_{pke}(pw)$, the simulator can decrypt using ske and obtain the password guess pw. However, as we will now show, this seems not to suffice.

In order to see where the difficulty arises, consider an ideal-model adversary/simulator $\mathcal{S}$ that has access to the functionality $\mathcal{F}_{\mathsf{pwKE}}$ and needs to simulate the KOY/GL protocol for a real-life adversary $\mathcal{A}$. Informally, simulating the server when the adversary impersonates a *client* can be carried out as follows: The simulator decrypts the ciphertext c_1 generated by the adversary and recovers the adversary's "password guess" pw (this decryption can be carried out because $\mathcal{S}$ chooses the common reference string so that it has the corresponding secret key). The simulator then sends pw to $\mathcal{F}_{\mathsf{pwKE}}$ as its own guess. If the guess is incorrect, then as described above, the smoothness of the hash function causes the honest parties to output independent random keys (as required in the ideal model with an interrupted session). In contrast, if the guess is correct then the simulator has learned the correct password and can continue the remainder of the execution exactly as an honest party would when using that password. However, consider what happens when the adversary impersonates a *server*. Here, the simulator must send some c_1 (presumably an encryption of some password pw') *before* the adversary replies with c_2. As before, the simulator can decrypt c_2, recover the password pw in it, and submit this guess to $\mathcal{F}_{\mathsf{pwKE}}$. However, if it turns out that pw is a *correct* guess, the simulator is stuck with a ciphertext c_1 that in all likelihood is an encryption of the wrong password. Not knowing the hashing key hk that $\mathcal{A}$ holds, the simulator cannot predict the value $H_{hk}(c_1, pw)$ that $\mathcal{A}$ will compute (since $(c_1, pw) \notin L$). Thus, the simulator seems to have no way of ensuring that the secret key that $\mathcal{A}$ computes is the same as the one that the environment gets from the functionality (via the client).[8]

To overcome this problem, we modify the protocol by having the server send a "pre-flow" c_0 which also contains an encryption of the password; i.e., $c_0 = E(pw; r_0)$. Then, when the server later sends c_2, it proves in zero knowledge that c_0 and c_2 are encryptions of the same value. We stress that the session-key is still computed using only c_1 and c_2 and so it is important that consistency hold between these two only (where by consistency, we mean that they are both an encryption of the same password). The modified protocol is sketched in Figure 4. (Note that we switch the "client" and "server" roles so that the client is still the one who sends the first message.)

We now describe the high-level simulation strategy for the modified protocol (and thus why this modification solves the above-described problem):

---

probability of successfully guessing the password, even in the case that related and partially-leaked passwords are used.

8 This problem does not arise in the proofs of KOY/GL, since the "simulator" there can just halt and the adversary is declared successful. Our simulator, on the other hand, must continue to simulate both when the adversary fails *and* when it succeeds.

P_i (client)		P_j (server)
	CRS: pke, pke'	
$c_0 \leftarrow E'_{pke'}(pw; r_0)$	$\xrightarrow{c_0}$	
		$(sk, vk) \leftarrow \text{sigKey}(\$)$
	$\xleftarrow{c_1, vk}$	$c_1 \leftarrow E_{pke}(pw; r_1)$
$c_2 \leftarrow E'_{pke'}(pw, r_2)$		
$hk \leftarrow \mathcal{H}$		
$hp \leftarrow \alpha(hk; c_1)$	$\xrightarrow{c_2, hp}$	
	$\xrightarrow{\text{ZKP}_{(c_0 \approx c_2)}}$	
		$hk' \leftarrow \mathcal{H}'$
		$hp' \leftarrow \alpha'(hk'; c_2)$
	$\xleftarrow{hp', \sigma}$	$\sigma \leftarrow \mathsf{Sign}_{sk}(c_2, hp, hp')$
if $(\mathsf{Verify}_{vk}((c_2, hp, hp'), \sigma) = 1)$		
session-key $\leftarrow H_{hk}(c_1, pw)$		session-key $\leftarrow h_{hp}(c_1, pw; r_1)$
$+ h'_{hp'}(c_2, pw; r_2)$		$+ H'_{hk'}(c_2, pw)$

Fig. 4. The core of the universally-composable protocol.

1. *Case 1 — the adversary $\mathcal{A}$ impersonates the client:* The simulator $\mathcal{S}$ obtains the ciphertext c_0, decrypts it to obtain pw and sends pw to $\mathcal{F}_{\mathsf{pwKE}}$ as the password guess. If this guess is correct, then $\mathcal{S}$ continues the simulation using the same pw and consistency is achieved. Note that the zero-knowledge proof ensures that the ciphertext c_2 later sent by $\mathcal{A}$ is also an encryption of pw (therefore in this case, consistency between c_0 and c_1 implies consistency between c_1 and c_2, as required).

2. *Case 2 — the adversary $\mathcal{A}$ impersonates the server:* In this case, the simulator $\mathcal{S}$ generates the pre-flow ciphertext c_0 as an encryption of some default value (which actually will not be any password). Then, upon receiving c_1 from $\mathcal{A}$, the simulator $\mathcal{S}$ decrypts it to obtain pw and sends pw to $\mathcal{F}_{\mathsf{pwKE}}$ as the password guess. If the guess is correct, then $\mathcal{S}$ generates c_2 to also be an encryption of pw. Notice that c_1 and c_2 are now consistent in that they both encrypt the correct password pw. The only problem remaining in the simulation is that $\mathcal{S}$ is supposed to prove that c_0 and c_2 are indeed consistent (which in this case they are not). It does this by using the *zero-knowledge simulator* for the zero-knowledge proof of consistency. By the zero-knowledge property, this proof is indistinguishable from a real one (and this holds even though the statement in this case is false). We therefore conclude that in the case of a correct password guess, consistency is achieved and in addition, the adversary cannot distinguish its view in the simulation from its view in a real execution.

We stress that Figure 4 is only a sketch of the protocol and does not contain all the details. For example, the full protocol uses *labeled encryption* [29] in order to bind certain protocol information to the ciphertexts (such as the session-id and the verification key of the signature scheme) and in order to prevent other types of man-in-the-middle attacks. (Labels were used implicitly for the same purpose in [24, 18].) A detailed description of the protocol and its proof can be found in the full version, where we also provide a formal statement and proof of the following result:

Theorem 1 (main theorem – informally stated): *Assume the existence of CCA-secure encryption schemes with smooth projective hash functions, and simulation-sound zero-knowledge proofs. Then there exists a protocol in the common reference string model that securely realizes the $\mathcal{F}_{\mathsf{pwKE}}$ functionality in the presence of static-corruption adversaries.*

We remark that all the building blocks of our protocol can be built under the DDH, quadratic residuosity, or N^{th}-residuosity assumptions, so UC-secure password-based key exchange is possible under any of these assumptions. For the most efficient instantiation, we would use encryption and smooth projective hash proofs based on DDH, a collision-resistant hash function for the one-time signature, and a simulation-sound zero-knowledge proof system [17, 25] based on strong RSA. Hence, the end result would rely on all of these assumptions for its security.

Efficiency notes. Considering the protocol as depicted in Figure 4, we emphasize that it suffices to use a (simulation sound) zero-knowledge proof of membership, rather than a proof of knowledge. This allows for efficient solutions; see [17]. Furthermore, it suffices to generate c_0 and c_2 using an encryption scheme that is only *CPA-secure*, rather than CCA-secure.[9] Thus, the encryption scheme E in Figure 4 (that is used to generate c_1) is CCA-secure, but the encryption scheme E' (that is used to generate c_0 and c_2) is only CPA-secure. Using a CPA-secure scheme for E' provides efficiency improvements in the encryption itself, the projective hashing, and the proof of consistency. In [18] a highly efficient and simple construction of smooth projective hashing was demonstrated for the El Gamal encryption scheme (which is CPA-secure under the DDH assumption). Furthermore, proving consistency of El Gamal encryptions is more efficient than proving consistency of, e.g., CCA-secure Cramer-Shoup encryptions.

4 Additional Results

We describe some additional results that appear in the full version of the paper.

[9] In fact, the second encryption in the KOY/GL protocols can be generated using a CPA-secure scheme (e.g., El Gamal) as well; see [23]. This yields the most efficient known password-based key-exchange protocol in the standard model (i.e., without random oracles), albeit under a weaker definition than the one considered here.

Functionality $\mathcal{F}_{\mathsf{pwSC}}$

The functionality $\mathcal{F}_{\mathsf{pwSC}}$ is parameterized by a security parameter k. It interacts with an adversary S and a set of parties via the following queries.

Upon receiving a $(\mathsf{NewSession}, sid, P_i, P_j, pw, \mathrm{role})$ **query from party** P_i**:**
Send $(\mathsf{NewSession}, sid, P_i, P_j, \mathrm{role})$, to S. In addition, if this is the first $\mathsf{NewSession}$ query, or if this is the second $\mathsf{NewSession}$ query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark it fresh.

Upon receiving a $(\mathsf{TestPwd}, sid, P_i, pw')$ **query from the adversary** S**:**
If there is a record of the form (P_i, P_j, pw) which is fresh, then do: If $pw = pw'$, then mark the record $\mathsf{compromised}$ and reply to S with "correct guess". If $pw \neq pw'$, then mark it $\mathsf{interrupted}$ and reply with "wrong guess".

Upon receiving a (Send, sid, m) **query from** P_i**:**
If there is a record of the form (P_i, P_j, pw) then:
- If this record is $\mathsf{compromised}$, or P_i is corrupted, then send (sid, P_i, m) to the adversary.
- If this record is fresh, and there is a record (P_j, P_i, pw') with $pw' = pw$ that is also fresh, then send $(\mathsf{Received}, sid, m)$ to P_j and $(sid, P_i, |m|)$ to the adversary.
- In any other case, send $(sid, P_i, |m|)$ to the adversary.

Upon receiving an $(\mathsf{ExpireSession}, sid)$ **query from** P_i**:**
If there is a record of the form (P_i, P_j, pw), it marks the record as $\mathsf{expired}$.

Fig. 5. The password-based secure channels functionality, $\mathcal{F}_{\mathsf{pwSC}}$

Password-based secure channels. Arguably, the typical use of a key-exchange protocol is the establishment of *secure channels* that enable the parties to communicate privately and reliably. The case of password-based key-exchange is no exception. In Figure 5, we describe the password-based secure channels functionality $\mathcal{F}_{\mathsf{pwSC}}$. In the full version, we show that our definition of password-based key-exchange suffices for securely realizing this functionality, thus providing additional "justification" for our definition of $\mathcal{F}_{\mathsf{pwKE}}$. The definition of $\mathcal{F}_{\mathsf{pwSC}}$ is analogous to the password-based key-exchange functionality. Notice that if two parties have sent $\mathsf{NewSession}$ queries with the same identifiers and passwords, *and* the adversary has not guessed this password or interrupted the session, then the functionality faithfully passes messages from the first party to the second. Furthermore, the adversary learns only the length of the message sent. Thus, the functionality provides *reliable and private communication*, as desired. (The functionality only deals with unidirectional communication from P_i to P_j; it can be repeated in order to obtain bidirectional communication.)

Impossibility of realizing $\mathcal{F}_{\mathsf{pwKE}}$ in the plain model. Our protocol is cast in the common reference string model which assumes some (albeit, rather minimal) trusted setup phase. An important question to ask is whether or not this is *necessary* for obtaining security. In the full version of this paper, we prove that the $\mathcal{F}_{\mathsf{pwKE}}$ functionality cannot be securely realized in the "plain" model (i.e., without using a common reference string or some other augmentation to the basic model). Our proof is similar to the proofs of impossibility in [10]. The basic idea is as follows. Consider an environment that internally runs the code of one of the honest parties. The ideal-model simulator for such an environment must succeed while interacting with it in the same way that real parties interact. (This holds by the definition of the UC framework which requires a black-box simulator which cannot rewind.) Now, if simulation can be carried out in such a scenario, then it can also be carried out while interacting with a real honest party (because the specific environment we have chosen behaves like an honest party). This means that anything the ideal-model simulator/adversary can do with respect to the environment, a real-model adversary can do with respect to an honest party. In particular, in order for the simulation to succeed, the ideal-model simulator must be able to set the output session-key to be the same as the key output by the ideal functionality. Thus, a real-model adversary can also do this, in contradiction to the security requirements of the key-exchange protocol.

References

1. B. Barak, Y. Lindell, and T. Rabin, Protocol Initialization for the Framework of Universal Composability. Manuscript. Available from the ePrint archive, report 2004/006 from `http://eprint.iacr.org`.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. *Advances in Cryptology – Crypto 1998*, LNCS vol. 1462, Springer-Verlag, pp. 26–45, 1998
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. *Advances in Cryptology – Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 139–155, 2000.
4. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *Advances in Cryptology – Crypto 1993*, LNCS vol. 773, Springer-Verlag, pp. 232–249, 1993.
5. S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. Proc. *IEEE Security and Privacy*, IEEE, pp. 72–84, 1992.
6. V. Boyko. On All-or-Nothing Transforms and Password-Authenticated Key Exchange. PhD thesis, MIT, EECS department, 2000.
7. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password Authentication and Key Exchange Using Diffie-Hellman. *Advances in Cryptology – Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 156–171, 2000.
8. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 136–145, 2001.
9. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. *Advances in Cryptology – Eurocrypt 2002*, LNCS vol. 2332, Springer-Verlag, pp. 337–351, 2002.
10. R. Canetti, E. Kushilevitz and Y. Lindell. On the Limitations of Universal Composable Two-Party Computation Without Set-Up Assumptions. In *EUROCRYPT 2003,* Springer-Verlag (LNCS 2656), pages 68–86, 2003.
11. R. Canetti and T. Rabin. Universal Composition with Joint State. *Advances in Cryptology – Crypto 2003*, LNCS vol. 2729, Springer-Verlag, pp. 265–281, 2003.

12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. *Advances in Cryptology – Crypto 1994*, LNCS vol. 839, Springer-Verlag, pp. 174–187, 1994.
13. R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. *Advances in Cryptology – Eurocrypt 2002*, LNCS vol. 2332, Springer-Verlag, pp. 45–64, 2002.
14. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust Non-Interactive Zero-Knowledge. *Advances in Cryptology – Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 566–598, 2001.
15. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM J. Computing* 30(2): 391–437, 2000.
16. S. Even, O. Goldreich, and S. Micali. On-Line/Off-Line Digital Signatures. *J. Cryptology* 9(1):35-67, 1996.
17. J. Garay, P. MacKenzie, and K. Yang. Strengthening Zero-Knowledge Protocols Using Signatures. *Advances in Cryptology – Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 177–194, 2003.
18. R. Gennaro and Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. *Advances in Cryptology – Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 524–543, 2003.
19. O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only. *Advances in Cryptology – Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 408–432, 2001.
20. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences* 28:270–299, 1984.
21. L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5): 648–656, 1993.
22. Shai Halevi and Hugo Krawczyk. Public-Key Cryptography and Password Protocols. *ACM Trans. on Information and Systems Security,* 2(3):230–268, 1999.
23. J. Katz, P. MacKenzie, G. Taban, and V. Gligor. Two-Server Password-Only Authenticated Key Exchange. Manuscript, Jan. 2005.
24. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. *Advances in Cryptology – Eurocrypt 2001*, LNCS vol. 2045, Springer-Verlag, pp. 475–494, 2001.
25. P. MacKenzie and K. Yang. On Simulation-Sound Trapdoor Commitments. *Advances in Cryptology – Eurocrypt 2004*, LNCS vol. 3027, Springer-Verlag, pp. 382–400, 2004. Available from the ePrint archive, report 2003/252 from `http://eprint.iacr.org`.
26. M.H. Nguyen and S. Vadhan. Simpler Session-Key Generation from Short Random Passwords. Proceedings of the *1st Theory of Cryptography Conference* (TCC'04), LNCS vol. 2951, Springer-Verlag, pp. 442–445, 2004.
27. C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. *Advances in Cryptology – Crypto 1991*, LNCS vol. 576, Springer-Verlag, pp. 433–444, 1991.
28. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 543–553, 1999.
29. V. Shoup. A Proposal for an ISO Standard for Public Key Encryption (version 2.1). December, 2001. Available from the ePrint archive, report 2001/112 from `http://eprint.iacr.org`.