The extended abstract of this work appears in [2]. This is the full version.

Password-Based Authenticated Key Exchange in the Three-Party Setting

Michel Abdalla   Pierre-Alain Fouque   David Pointcheval

Departement d'Informatique, École normale supérieure
45 Rue d'Ulm, 75230 Paris Cedex 05, France
`{Michel.Abdalla,Pierre-Alain.Fouque,David.Pointcheval}@ens.fr`
`http://www.di.ens.fr/users/{mabdalla,fouque,pointche}`.

Abstract

Password-based authenticated key exchange (PAKE) are protocols which are designed to be secure even when the secret key used for authentication is a human-memorable password. In this paper, we consider PAKE protocols in the three-party scenario, in which the users trying to establish a common secret do not share a password between themselves but only with a trusted server. Towards our goal, we recall some of the existing security notions for PAKE protocols and introduce new ones that are more suitable to the case of generic constructions of three-party protocols. We then present a natural generic construction of a three-party PAKE protocol from any two-party PAKE protocol and prove its security. To the best of our knowledge, the new protocol is the first provably-secure PAKE protocol in the three-party setting.

Keywords: password, authenticated key exchange, key distribution, multi-party protocols.

1 Introduction

Motivation. A fundamental problem in cryptography is how to communicate securely over an insecure channel, which might be controlled by an adversary. It is common in this scenario for two parties to encrypt and authenticate their messages in order to protect the privacy and authenticity of these messages. One way of doing so is to use public-key encryption and signatures, but the cost associated with these primitives may be too high for certain applications. Another way of addressing this problem is for users to first establish a common secret key via a key exchange protocol and then use this key to derive keys for symmetric encryption and message authentication schemes.

In practice, one finds several flavors of key exchange protocols, each with its own benefits and drawbacks. Among the most popular ones is the 3-party *Kerberos* authentication system [26]. Another one is the 2-party SIGMA protocol [20] used as the basis for the signature-based modes of the Internet Key Exchange (IKE) protocol. Yet another flavor of key exchange protocols which has received significant attention recently are those based on passwords.

Password-based key exchange. Password-based key exchange protocols assume a more realistic scenario in which secret keys are not uniformly distributed over a large space, but rather chosen from a small set of possible values (a four-digit pin, for example). They also seem more convenient since human-memorable passwords are simpler to use than, for example, having additional cryptographic devices capable of storing high-entropy secret keys. The vast majority of protocols found in practice do not account, however, for such scenario and are often subject to so-called *dictionary* attacks. Dictionary attacks are attacks in which an adversary tries to break the security of a scheme by a brute-force method, in which it tries all possible combinations of secret keys in a given small set of values (i.e., the dictionary). Even though these attacks are not very effective in the case of high-entropy keys, they can be very damaging when the secret key is a password since the attacker has a non-negligible chance of winning. Such attacks are usually divided in two categories: *off-line* and *online* dictionary attacks.

To address this problem, several protocols have been designed to be secure even when the secret key is a password. The goal of these protocols is to restrict the adversary's success to on-line guessing attacks only. In these attacks, the adversary must be present and interact with the system in order to be able to verify whether its guess is correct. The security in these systems usually relies on a policy of invalidating or blocking the use of a password if a certain number of failed attempts has occurred.

Password-based key exchange in the 3-party setting. Passwords are mostly used because they are easier to remember by humans than secret keys with high entropy. Consequently, users prefer to remember very few passwords but not many. However, in scenarios where a user wants to communicate with many other users, then the number of passwords that he or she would need to remember would be linear in the number of possible partners. In order to limit the number of passwords that each user needs to remember, we consider in this paper password-based authenticated key exchange in the 3-party model, where each user only shares a password with a trusted server. The main advantage of this solution is that it provides each user with the capability of communicating securely with other users in the system while only requiring it to remember a single password. This seems to be a more realistic scenario in practice than the one in which users are expected to share multiple passwords, one for each party with which it may communicate privately. Its main drawback is that the server is needed during the establishment of all communication as in the Needham and Schroeder protocol [23].

Key privacy. One potential disadvantage of a 3-party model is that the privacy of the communication with respect to the server is not always guaranteed. Since we want to trust as little as possible the third party, we develop a new notion called key privacy which roughly means that, even though the server's help is required to establish a session key between two users in the system, the server should not be able to gain any information on the value of that session key. Here we assume that the server is honest but curious. Please note that key distribution schemes usually do *not* achieve this property.

Insider attacks. One of the main differences between the 2-party and the 3-party scenarios is the existence of insider attacks. To better understand the power of these attacks, consider the protocol in Figure 1, based on the encrypted key exchange of Bellovin and Merritt [11], in which the server simply decrypts the message it receives and re-encrypts it under the other user's password. In this protocol, it is easy to see that one can mount an off-line dictionary by simply playing the role of one of the involved parties. Notice that both A and B can obtain the necessary information to mount an off-line dictionary attack against each other simply by eavesdropping on the messages that are sent out by the server. More specifically, A and B can respectively learn the values $X_S^\star = \mathcal{E}_{PW_B}(X_S)$ and $Y_S^\star = \mathcal{E}_{PW_A}(Y_S)$ and mount a dictionary attack against each other using the fact that $X_S = X_A$ and $Y_S = Y_B$. Despite being also possible in the case of 2-party protocols (see [12]), insider attacks seem easier to mount in the 3-party scenario and thus must be taken into account.

Public information: $\mathbb{G}, g, p, \mathcal{E}, \mathcal{D}, H$

$$\begin{array}{ccccc}
\text{Client A} & & \text{Server} & & \text{Client B}\\
pw_A \in \mathsf{Password} & & pw_A, pw_B \in \mathsf{Password} & & pw_B \in \mathsf{Password}\\
 & & & & \\
x \xleftarrow{R} \mathsf{Z}_p \; ; \; X_A \leftarrow g^x & & & & y \xleftarrow{R} \mathsf{Z}_p \; ; \; Y_B \leftarrow g^y\\
X_A^\star \leftarrow \mathcal{E}_{pw_A}(X_A) & & & & Y_B^\star \leftarrow \mathcal{E}_{pw_B}(Y_B)\\
 & \xrightarrow{X_A^\star} & & \xleftarrow{Y_B^\star} & \\
 & & X_S \leftarrow \mathcal{D}_{pw_A}(X_A^\star) & & \\
 & & Y_S \leftarrow \mathcal{D}_{pw_B}(Y_B^\star) & & \\
 & & Y_S^\star \leftarrow \mathcal{E}_{pw_A}(Y_S) & & \\
 & & X_S^\star \leftarrow \mathcal{E}_{pw_B}(X_S) & & \\
 & \xleftarrow{Y_S^\star} & & \xrightarrow{X_S^\star} & \\
Y_A \leftarrow \mathcal{D}_{pw_A}(Y_S^\star) & & & & X_B \leftarrow \mathcal{D}_{pw_B}(X_S^\star)\\
K_A \leftarrow Y_A^x & & & & K_B \leftarrow X_B^y\\
SK_A \leftarrow H(A \,\|\, B \,\|\, S \,\|\, K_A) & & & & SK_B \leftarrow H(A \,\|\, B \,\|\, S \,\|\, K_B)
\end{array}$$

Figure 1: A 3-party password-based encrypted key exchange protocol that is insecure against insider attacks. $\mathcal{E}_{pw}$ and $\mathcal{D}_{pw}$ represent, respectively, the encryption and decryption algorithms of a cipher, such as AES [3], using the password pw as the encryption and decryption key.

A NEW SECURITY MODEL. In order to analyze the security of 3-party password-based authenticated key exchange protocols, we put forward a new security model and define two notions of security: indistinguishability of the session key and key privacy with respect to the server. The first of these notions is the usual one and is a straight-forward generalization of the equivalent notion in the 2-party password-based authenticated key exchange model. The second one is new and particular to the new setting, and captures the privacy of the key with respect to the trusted server to which all passwords are known.

A GENERIC CONSTRUCTION. In this paper, we consider a generic construction of a 3-party password-based protocol. Our construction is a natural one, building upon existing 2-party password-based key exchange and 3-party symmetric key distribution schemes, to achieve provable security in the strongest sense. Moreover, our construction is also modular in the sense that it can be broken into two parts, a 3-party password-based key distribution protocol and 2-party authenticated key exchange. The second part is only needed if key privacy with respect to the server is required.

THE NEED FOR NEW SECURITY NOTIONS. Surprisingly, the proof of security for the new scheme *does not* seem to follow from the usual security notions for the underlying schemes as one would expect and seems to require a *new* and *stronger* notion of security for the underlying 2-party password-based scheme (see Section 2). In fact, this new security notion is not specific to password-based schemes and is one of the main contributions of this paper. Fortunately, we observe that most existing 2-party password-based schemes do in fact satisfy this new property [14, 17, 19, 22]. More specifically, only a few small changes are required in their proof in order to achieve security in the new model. The bounds obtained in their proof remain essentially unchanged.

Contributions. In this paper, we consider password-based key exchange with implicit authentication in the 3-party model, where each user only shares a password with a trusted server.

New security models. Towards our goal, we put forth a new formal security model that is appropriate for the 3-party password-based authenticated key exchange scenario and give precise definitions of what it means for it to be secure. Our model builds upon those of Bellare and Rogaway [9, 10] for key distribution schemes and that of Bellare, Pointcheval, and Rogaway [7] for password-based authenticated key exchange.

New security notions. We also present a new and stronger model for 2-party authenticated key exchange protocols, which we call the Real-Or-Random model. Our new model is provably stronger than the existing model, to which we refer to as the Find-Then-Guess model, in the sense that a scheme proven secure in the new model is also secure in the existing model. However, the reverse is not necessarily true due to an unavoidable non-constant factor loss in the reduction. Such losses in the reduction are extremely important in the case of password-based protocols. In addition to the indistinguishability of the session key, we present a new property, called key privacy, which is specific to 3-party key exchange protocols. This new notion captures in a quantitative way the idea that the session key shared between two instances should only be known to these two instances and no one else, including the trusted server.

A generic construction in the standard model. We present a generic and natural framework for constructing a 3-party password-based authenticated key exchange protocol from any secure 2-party password-based one. We do so by combining a 3-party key distribution scheme, an authenticated Diffie-Hellman key exchange protocol, and the 2-party password-based authenticated key exchange protocol. The proof of security relies solely on the security properties of underlying primitives it uses and does not assume the Random Oracle model [8]. Hence, when appropriately instantiated, this construction yields a secure protocol in the standard model.

Related Work. Password-based authenticated key exchange has been extensively studied in the last few years, in various environments. The seminal work in this area is the encrypted key exchange protocol by Bellovin and Merritt [11], in which two users execute an encrypted version of the Diffie-Hellman key exchange protocol [16]. In their protocol, each flow is encrypted using the password shared between these two users as the symmetric key. Unfortunately, due to the lack of a proper security model, no formal security analysis was presented for their protocol.

The first formal security model for authenticated key exchange protocols between two parties was introduced by Bellare and Rogaway [9]. The latter has been extended to the password-based setting [7, 13], with security analyses of the above 2-party password-based key exchange, under idealized assumptions, such as the random oracle and the ideal cipher models. Password-based schemes, provably secure in the standard model, have been recently proposed [19, 18, 17], but only for two parties. Only few papers [15, 21, 27] have considered password-based protocols in the 3-party setting, but none of their schemes enjoys provable security. In fact, our generic construction seems to be the first provably-secure 3-party password-based authenticated key exchange protocol.

Another related line of research is authenticated key exchange *in the* 3*-party setting*. The first work in this area is the protocol of Needham and Schroeder [23], which inspired the *Kerberos* distributed system [26]. Later, Bellare and Rogaway introduced a formal security model in this scenario along with a construction of the first provably-secure symmetric-key-based key distribution scheme [10]. In this paper, we consider the special but important case in which the secret keys are drawn from a small set of values.

Organization. In Section 2, we recall the existing security model for 2-party password-based authenticated key exchange and introduce a new one. Next, in Section 3, we introduce new models for 3-party password-based authenticated key exchange. Then, in Section 4, we relate the new security notions being introduced in this paper to some of the existing ones. Section 5 then presents our generic construction of a 3-party password-based authenticated key exchange protocol, called GPAKE, along with the security claims and suggestions on how to instantiate it. The proofs of security for GPAKE are given in Section 6. We conclude this paper by presenting some future extensions of this work in Section 7.

2 Security models for 2-party password-based key exchange

A secure 2-party password-based key exchange (2PAKE) is a protocol where the parties use their password in order to derive a common session key sk that will be used to build secure channels. Loosely speaking, such protocols are said to be secure against *dictionary attacks* if the advantage of an attacker in distinguishing a real session key from a random key is less than $O(n/|\mathcal{D}|) + \varepsilon(k)$ where $|\mathcal{D}|$ is the size of the dictionary $\mathcal{D}$, n is the number of active sessions and $\varepsilon(k)$ is a negligible function depending on the security parameter k.

In this section, we recall the security model for 2-party password-based authenticated key exchange protocols introduced by Bellare, Pointcheval, and Rogaway (BPR) [7] and introduce a new one. For reasons that will soon become apparent, we refer to the new model as the Real-Or-Random (ROR) model and to the BPR model as the Find-Then-Guess (FTG) model, following the terminology of Bellare *et al.* for symmetric encryption schemes [5].

2.1 Communication model

Protocol participants. Each participant in the 2-party password-based key exchange is either a client $C \in \mathcal{C}$ or a server $S \in \mathcal{S}$. The set of all users or participants $\mathcal{U}$ is the union $\mathcal{C} \cup \mathcal{S}$.

Long-lived keys. Each client $C \in \mathcal{C}$ holds a password pw_C. Each server $S \in \mathcal{S}$ holds a vector $pw_S = \langle pw_C \rangle_{C \in \mathcal{C}}$ with an entry for each client. pw_C and pw_S are also called the long-lived keys of client C and server S.

Protocol execution. The interaction between an adversary A and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. During the execution, the adversary may create several concurrent instances of a participant. These queries are as follows, where U^i denotes the instance i of a participant U:

- *Execute*(C^i, S^j): This query models passive attacks in which the attacker eavesdrops on honest executions between a client instance C^i and a server instance S^j. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.

- *Send*(U^i, m): This query models an active attack, in which the adversary may intercept a message and then either modify it, create a new one, or simply forward it to the intended participant. The output of this query is the message that the participant instance U^i would generate upon receipt of message m.

2.2 Security definitions

Partnering. As in [7], we use the notion of partnering based on session identifiers (*sid*) and partner identifiers (*pid*). More specifically, let the session identifier of a client or server instance be a function of all the messages sent and received by that instance as specified by the key exchange protocol. Let the partner identifier of a client or server instance be the instance with which a common secret key is to be established. Then, two instances U_1^i and U_2^j are said to be partners if the following conditions are met: (1) Both U_1^i and U_2^j accept; (2) Both U_1^i and U_2^j share the same session identifiers; (3) The partner identifier for U_1^i is U_2^j and vice-versa; and (4) No instance other than U_1^i and U_2^j accepts with a partner identifier equal to U_1^i or U_2^j. In practice, the *sid* can be taken to be the partial transcript of the conversation between the client and the server instances before the acceptance.

Freshness. In order to properly formalize security notions for the session key, one has to be careful to avoid cases in which adversary can trivially break the security of the scheme. For example, an adversary who is trying to distinguish the session key of an instance U^i from a random key can trivially do so if it obtains the key for that instance through a *Reveal* query (see definition below) to instance U^i or its partner. Instead of explicitly defining a notion of freshness and mandating the adversary to only perform tests on *fresh* instances as in previous work, we opted to embed that notion inside the definition of the oracles.

Indistinguishability in the Find-Then-Guess model. This is the definition currently being used in the literature. In order to measure the indistinguishability of the session key of user instance, the adversary is given access to two additional oracles: the *Reveal* oracle, which models the misuse of session keys by a user, and the *Test* oracle, which tries to capture the adversary's ability (or inability) to tell apart a real session key from a random one. Let b be a bit chosen uniformly at random at the beginning of the experiment defining indistinguishability in the Find-Then-Guess (FTG) model. These oracles are defined as follows.

- *Reveal*(U^i): If a session key is not defined for instance U^i or if a *Test* query was asked to either U^i or to its partner, then return $\perp$. Otherwise, return the session key held by the instance U^i.

- *Test*(U^i): If no session key for instance U^i is defined or if a *Reveal* query was asked to either U^i or to its partner, then return the undefined symbol $\perp$. Otherwise, return the session key for instance U^i if $b = 1$ or a random key from the same domain if $b = 0$.

The adversary in this case is allowed to ask multiple queries to the *Execute*, *Reveal*, and *Send* oracles, but it is restricted to ask only a *single* query to the *Test* oracle. The goal of the adversary is to guess the value of the hidden bit b used by the *Test* oracle. The adversary is considered successful if it guesses b correctly.

Let Succ denote the event in which the adversary is successful. The **ftg-ake-advantage** of an adversary $\mathcal{A}$ in violating the indistinguishability of the protocol P in the FTG sense and the **advantage function** of the protocol P, when passwords are drawn from a dictionary $\mathcal{D}$, are respectively

$$\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}(\mathcal{A}) = 2 \cdot \Pr[\,\textsc{Succ}\,] - 1 \quad \text{and} \quad \mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}(t, R) = \max_{\mathcal{A}}\{\, \mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}(\mathcal{A})\,\},$$

where the maximum is over all $\mathcal{A}$ with time-complexity at most t and using resources at most R (such as the number of queries to its oracles). The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the experiments defining the security plus the code size of the adversary [1]. Note that the advantage of an adversary that simply guesses the bit b is 0 in the above definition due to the rescaling of the probabilities.

We say a 2-party password-based key exchange protocol P is secure in the FTG sense if the advantage $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}$ is only negligibly larger than $kn/|\mathcal{D}|$, where n is number of active sessions and k is a constant. A session is said to be active if it involves *Send* queries by the adversary. We note that $k = 1$ in the best scenario one can expect since an adversary that simply guesses the password and plays the role of a given user via *Send* queries has an advantage of $n/|\mathcal{D}|$.

In order to guarantee that security definitions are actually meaningful in practice, we assume henceforth that the size of the key space from which the session key is sampled is at least super-polynomial in the security parameter.

Indistinguishability in the Real-Or-Random model. This is a new definition. In the Real-Or-Random (ROR) model, we only allow the adversary to ask *Execute*, *Send*, and *Test* queries. In other words, the *Reveal* oracle that exists in the FTG model is no longer available to the adversary. Instead, we allow the adversary to ask as many *Test* queries as it wants to different instances. All *Test* queries in this case will be answered using the same value for the hidden bit b that was chosen at the beginning . That is, the keys returned by the *Test* oracle are either all real or all random. However, in the random case, the same random key value should be returned for *Test* queries that are asked to two instances which are partnered. Please note that the *Test* oracle is the oracle modeling the misuse of keys by a user in this case. The goal of the adversary is still the same: to guess the value of the hidden bit b used to answer *Test* queries. The adversary is considered successful if it guesses b correctly.

Let $\textsc{Succ}$ denote the event in which the adversary is successful. The **ror-ake-advantage** of an adversary $\mathcal{A}$ in violating the indistinguishability of the protocol P in the ROR sense, $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ror-ake}}(\mathcal{A})$, and the **advantage function** $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ror-ake}}(t, R)$ of the protocol P are then defined as in the previous definition.

As in FTG case, we say a 2-party password-based key exchange protocol P is secure in the ROR sense if the advantage $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}$ is only negligibly larger than $kn/|\mathcal{D}|$, where n is number of active sessions and k is a constant.

Relation between notions. As we prove in Section 4, the Real-Or-Random (ROR) security model is actually stronger than the Find-Then-Guess (FTG) security model, assuming that constant factor losses in the security reduction are acceptable. More specifically, we show that proofs of security in the ROR model can be easily translated into proofs of security in the FTG model with only a 2 factor loss in the reduction (see Lemma 4.1). The reverse, however, is not necessarily true since the reduction is not security preserving. There is a loss of non-constant factor in the reduction (see Lemma 4.2). Moreover, the loss in the reduction cannot be avoided as there exist schemes for which we can prove such a loss in security exists (see Proposition 4.3).

To better understand the gap between the two notions, imagine a password-based scheme that was proven secure in the FTG model. By definition, the advantage of any adversary is at most $O(n/|\mathcal{D}|) + \varepsilon(k)$, where n is the number of active sessions and $\varepsilon(k)$ is a negligible term. By applying the reduction, we can show that no adversary can do better than $O(n^2/|\mathcal{D}|) + n \cdot \varepsilon(k)$, which is not enough to guarantee the security of the same scheme in the ROR model. Note that such a gap is not as important in the case where high-entropy keys are used since both terms in the expression would be negligible.

As a consequence, we cannot take for granted the security of the existing schemes and new proofs of security need be provided. Fortunately, we would like to point out here that the security proof for several of the existing schemes can be easily modified to meet the new security goals with essentially the same bounds. The reason for that is that the security proofs of most existing password-based schemes in fact prove something stronger than what is required by the security model. More specifically, most proofs generally show that not only the session key being tested looks random, but all the keys that may be involved in a reveal query also look random to an adversary that does not know the secret password, thus satisfying the security requirements of our new model. In particular, this is the case for the KOY protocol [19] and its generalization [17], and some other schemes based on the encrypted key exchange scheme of Bellovin and Merritt [11] (e.g., [14, 22]).

Since most existing password-based schemes do seem to achieve security in the new and stronger security model and since the latter appears to be more applicable to situations in which one wishes to use a password-based key exchange protocol as a black box, we suggest the use of our new model when proving the security of new password-based schemes.

Relation to simulation models. In [24], the Find-Then-Guess model of [10] is shown to be equivalent to simulation models in the sense that a scheme that is proven secure in one model is also secure in the other model. By closely examining their proof, one can easily see that the equivalence does not apply to the case of password-based protocols due to the non-security-preserving reduction. It seems, however, that their proof of equivalence can be adapted to show the equivalence between the simulation model and the Real-Or-Random model that we introduce in this paper in the case of password-based protocols.

3 Security models for 3-party password-based key exchange

In this section, we put forward new formal security models for 3-party password-authenticated key exchange and key distribution protocols. Our models are generalizations of the model of Bellare and Rogaway [10] for 3-party key distribution schemes to the password case and that of Bellare, Pointcheval, and Rogaway [7] for 2-party password-based authenticated key exchange.

3.1 Protocol Syntax

Protocol participants. Each participant in a 3-party password-based key exchange is either a client $U \in \mathcal{U}$ or a trusted server $S \in \mathcal{S}$. The set of clients $\mathcal{U}$ is made up of two disjoint sets: $\mathcal{C}$, the set of honest clients, and $\mathcal{E}$, the set of malicious clients. For simplicity, and without loss of generality [1], we assume the set $\mathcal{S}$ to contain only a single trusted server.

The inclusion of the malicious set $\mathcal{E}$ among the participants is one of the main differences between the 2-party and the 3-party models. Despite being also important in the 2-party model (see [12]), the inclusion of malicious users seems to be essential in the 3-party model as insider attacks appear to be a more realistic threat.

Long-lived keys. As in the 2-party case, each client $C \in \mathcal{C}$ holds a password pw_C and each server $S \in \mathcal{S}$ holds a vector $pw_S = \langle pw_C \rangle_{C \in \mathcal{C}}$ with an entry for each client. The only difference with respect to the 2-party case is that the set of passwords pw_E, where $E \in \mathcal{E}$, is assumed to be known by the adversary.

[1]This is because we are working in the concurrent model and because all servers are assumed to know the passwords of all users.

3.2 Communication model

The interaction between an adversary A and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. These queries are as follows:

- *Execute*$(U_1^{i_1}, S^j, U_2^{i_2})$: This query models passive attacks in which the attacker eavesdrops on honest executions among the client instances $U_1^{i_1}$ and $U_2^{i_2}$ and trusted server instance S^j. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.

- *SendClient*(U^i, m): This query models an active attack, in which the adversary may intercept a message and then modify it, create a new one, or simply forward it to the intended client. The output of this query is the message that client instance U^i would generate upon receipt of message m.

- *SendServer*(S^j, m): This query models an active attack against a server. It outputs the message that server instance S^j would generate upon receipt of message m.

3.3 Indistinguishability

The security definitions presented here build upon those of Bellare and Rogaway [9, 10] and that of Bellare, Pointcheval, and Rogaway [7].

NOTATION. Following [9, 10], we say an instance U^i has *accepted* if it goes into an accept mode after receiving the last expected protocol message.

PARTNERING. The definition of partnering in the 3-party setting is similar to the one given in the 2-party setting and is thus omitted here. We note, however, that, in order to guarantee that all participants in the same session end up with the same session identifier, the forwarding of messages may be required.

FRESHNESS. As in the 2-party case, we opted to embed the notion of freshness inside the definition of the oracles.

Indistinguishability in Find-Then-Guess model. This definition we give here is the straight-forward generalization of that of Bellare, Pointcheval, and Rogaway [7] for the 2-party case, combined with ideas of the model of Bellare and Rogaway [10] for 3-party key distribution. As in the 2-party case, we also define a *Reveal* oracle to model the misuse of session keys and a *Test* oracle to capture the adversary's ability to distinguish a real session key from a random one. Let b be a bit chosen uniformly at random at the beginning of the experiment defining indistinguishability in the FTG model. These oracles are defined as follows:

- *Reveal*(U^i): If a session key is not defined for instance U^i or if a *Test* query was asked to either U^i or to its partner, then return $\perp$. Otherwise, return the session key held by the instance U^i.

- *Test*(U^i): If no session key is defined for instance U^i or if the intended partner of U^i is part of the malicious set or if a *Reveal* query was asked to either U^i or to its partner, then return the invalid symbol $\perp$. Otherwise, return either the session key for instance U^i if $b = 1$ or a random key from the same domain if $b = 0$.

Consider an execution of the key exchange protocol P by an adversary $\mathcal{A}$, in which the latter is given access to the *Reveal*, *Execute*, *SendClient*, *SendServer*, and *Test* oracles and asks a single

Test query, and outputs a guess bit b'. Such an adversary is said to win the experiment defining indistinguishability if $b' = b$, where b is the hidden bit used by the *Test* oracle. Let Succ denote the event in which the adversary wins this game. The **ftg-ake-advantage** $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}(\mathcal{A})$ of an adversary $\mathcal{A}$ in violating the indistinguishability of the protocol P in the FTG sense and the **advantage function** $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}(t, R)$ of the protocol P are then defined as in previous definitions.

Like in the 2-party case, we say a 3-party password-based key exchange protocol P is secure in the FTG sense if the advantage $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ftg-ake}}$ is only negligibly larger than $kn/|\mathcal{D}|$, where n is number of active sessions and k is a constant.

Indistinguishability in Real-Or-Random model. This is a new definition. In the ROR model, *Reveal* queries are no longer allowed and are replaced by *Test* queries. In this case, however, the adversary is allowed to ask as many *Test* queries as it wants.

The modifications to the *Test* oracle are as follows. If a *Test* query is asked to a client instance that has not *accepted*, then return the invalid symbol $\perp$. If a *Test* query is asked to an instance of an honest client whose intended partner is dishonest or to an instance of a dishonest client, then return the real session key. Otherwise, the *Test* query returns either the real session key if $b = 1$ and a random one if $b = 0$, where b is the hidden bit selected at random prior to the first call. However, when $b = 0$, the same random key value should be returned for *Test* queries that are asked to two instances which are partnered. The goal of the adversary is still the same: to guess the value of the hidden bit used by the *Test* oracle. The adversary is considered successful if it guesses b correctly.

Consider an execution of the key exchange protocol P by an adversary A, in which the latter is given access to the *Execute*, *SendClient*, *SendServer*, and *Test* oracles, and outputs a guess bit b'. Such an adversary is said to win the experiment defining indistinguishability in the ROR sense if $b' = b$, where b is the hidden bit used by the *Test* oracle. Let Succ denote the event in which the adversary wins this game. The **ror-ake-advantage** $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ror-ake}}(\mathcal{A})$ of an adversary $\mathcal{A}$ in violating the indistinguishability of the protocol P in the ROR sense and the **advantage function** $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{ror-ake}}(t, R)$ of the protocol P are then defined as in previous definitions.

3.4 Key privacy with respect to the server

Differently from previous work, we define the notion of key privacy to capture, in a quantitative way, the idea that the session key shared between two instances should only be known to these two instances and no one else, including the trusted server. The goal of this new notion is to limit the amount of trust put into the server. That is, even though we rely on the server to help clients establish session keys between themselves, we still want to guarantee the privacy of these session keys with respect to the server. In fact, this is the main difference between a key distribution protocol (in which the session key is known to the server) and a 3-party key exchange protocol (for which the session key remains unknown to the server).

In defining the notion of key privacy, we have in mind a server which knows the passwords for all users, but that behaves in an honest but curious manner. For this reason, we imagine an adversary who has access to all the passwords as well as to the *Execute* and *SendClient* oracles but not to a *Reveal* oracle or to a *SendServer* oracle. The reason for not providing the adversary with a *SendServer* oracle is because the latter can be easily simulated by the adversary using the passwords. The reason for not providing the adversary with a a *Reveal* oracle is because, in the definition of key privacy, we only consider sessions in which the key is shared between two oracle instances. For these sessions, to capture the adversary's ability to tell apart the real session key from a random one from the same domain, we introduce a new type of oracle, called

TestPair. Let b is a bit chosen uniformly at random at the beginning of the experiment defining the notion of key privacy. The *TestPair* is defined as follows.

- *TestPair*(U_1^i, U_2^j): If client instances U_1^i and U_2^j do not share the same key, then return the invalid symbol $\perp$. Otherwise, return the real session key shared between client instances U_1^i and U_2^j if $b = 1$ or a random key from the same domain if $b = 0$.

Consider an execution of the key exchange protocol P by an adversary A in which the latter is given the passwords of all users and is allowed to ask multiple queries to its *Execute*, *SendClient*, and *TestPair* oracles. Let b' be its output. Such an adversary is said to win the experiment defining the key privacy if $b' = b$, where b is the hidden bit used by the *TestPair* oracle. Let Succ denote the event in which the adversary guesses b correctly. We can then define the **kp-advantage** $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{kp-ake}}(\mathcal{A})$ of $\mathcal{A}$ in violating the key privacy of the key exchange protocol P and the **advantage function** $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{kp-ake}}(t, R)$ of P as in previous definitions.

Finally, we say an adversary $\mathcal{A}$ succeeds in breaking the key privacy of a protocol P if its advantage $\mathbf{Adv}_{P,\mathcal{D}}^{\mathrm{kp-ake}}(\mathcal{A})$ is non-negligible.

4 Relations between notions

In this section, we prove the relation between the Find-Then-Guess (FTG) and Real-Or-Random (ROR) notions of security for authenticated key exchange protocols. The relation is not specific to password-based schemes, but its implications are more important in that scenario.

Lemma 4.1 Let $\mathbf{Adv}_{\mathsf{AKE}}^{\mathrm{ftg-ake}}(t, q_{\mathrm{send}}, q_{\mathrm{reveal}}, q_{\mathrm{exe}})$ represent the maximum advantage of an adversary in violating the indistinguishability of the protocol AKE in the FTG sense when the adversary has time-complexity at most t and asks at most q_{send} queries to its *Send* oracle, q_{reveal} queries to its *Reveal* oracle, and q_{exe} queries to its *Execute* oracle. Let $\mathbf{Adv}_{\mathsf{AKE}}^{\mathrm{ror-ake}}(t, q_{\mathrm{send}}, q_{\mathrm{test}}, q_{\mathrm{exe}})$ represent the maximum advantage of an adversary in violating the indistinguishability of the protocol AKE in the ROR sense when the adversary has time-complexity at most t and asks at most q_{send} queries to its *Send* oracle, q_{test} queries to its *Test* oracle, and q_{exe} queries to its *Execute* oracle. Then, for any protocol AKE, $\mathbf{Adv}_{\mathsf{AKE}}^{\mathrm{ftg-ake}}(t, q_{\mathrm{send}}, q_{\mathrm{reveal}}, q_{\mathrm{exe}}) \le 2 \cdot \mathbf{Adv}_{\mathsf{AKE}}^{\mathrm{ror-ake}}(t, q_{\mathrm{send}}, q_{\mathrm{test}}, q_{\mathrm{exe}})$, where $q_{\mathrm{test}} = q_{\mathrm{reveal}} + 1$.

Proof: In order to prove this lemma, we show how to build an adversary $\mathcal{A}_{\mathrm{ror}}$ against the indistinguishability of an authenticated key exchange protocol, AKE, in the ROR model given an adversary $\mathcal{A}_{\mathrm{ftg}}$ against the indistinguishability of the same protocol AKE in the FTG model. We know that $\mathcal{A}_{\mathrm{ftg}}$ has time-complexity at most t and that it asks at most q_{send}, q_{reveal}, and q_{exe} queries to its *Send*, *Reveal*, and *Execute* oracles, respectively.

The description of $\mathcal{A}_{\mathrm{ror}}$ is as follows. $\mathcal{A}_{\mathrm{ror}}$ starts by choosing a bit b uniformly at random and starts running $\mathcal{A}_{\mathrm{ftg}}$. If $\mathcal{A}_{\mathrm{ftg}}$ asks a *Send* query, then $\mathcal{A}_{\mathrm{ror}}$ asks the corresponding query to its *Send* oracle. If $\mathcal{A}_{\mathrm{ftg}}$ asks a *Execute* query, then $\mathcal{A}_{\mathrm{ror}}$ asks the corresponding query to its *Execute* oracle. If $\mathcal{A}_{\mathrm{ftg}}$ asks a *Reveal* query, then $\mathcal{A}_{\mathrm{ror}}$ asks a *Test* query to its *Test* oracle and uses the answer it receives as the answer to the *Reveal* query. If $\mathcal{A}_{\mathrm{ftg}}$ asks a *Test* query, then $\mathcal{A}_{\mathrm{ror}}$ asks the corresponding query to its *Test* oracle. If $b = 1$, then $\mathcal{A}_{\mathrm{ror}}$ uses the answer it received as the answer to the *Test* query. Otherwise, it returns a random key to $\mathcal{A}_{\mathrm{ftg}}$. Let b' be the final output of $\mathcal{A}_{\mathrm{ftg}}$. If $b' = b$, then $\mathcal{A}_{\mathrm{ror}}$ outputs 1. Otherwise, it outputs 0.

Note that $\mathcal{A}_{\mathrm{ror}}$ has time-complexity at most t and asks at most q_{send}, $q_{\mathrm{reveal}} + 1$, and q_{exe} queries to its *Send*, *Test*, and *Execute* oracles, respectively.

In order to analyze the advantage of $\mathcal{A}_{\text{ror}}$, first consider the case in which its *Test* oracle returns random keys. It is easy to see that, in this case, $\mathcal{A}_{\text{ftg}}$ cannot gain any information about the hidden bit b used to answer its single *Test* query. Therefore, the probability that $\mathcal{A}_{\text{ror}}$ outputs 1 is exactly $\frac{1}{2}$. Now consider the case in which its *Test* oracle returns the actual sessions keys. In this case, the simulation of *Reveal* is perfect and $\mathcal{A}_{\text{ror}}$ runs $\mathcal{A}_{\text{ftg}}$ exactly as in the experiment defining the indistinguishability of $\mathcal{A}_{\text{ftg}}$ in the FTG model. Therefore, the probability that $\mathcal{A}_{\text{ror}}$ outputs 1 is exactly $\frac{1}{2} + \frac{1}{2}\mathbf{Adv}_{\mathsf{AKE}}^{\text{ftg-ake}}(\mathcal{A}_{\text{ftg}})$ and, as a result, $\mathbf{Adv}_{\mathsf{AKE}}^{\text{ftg-ake}}(\mathcal{A}_{\text{ftg}}) \leq 2 \cdot \mathbf{Adv}_{\mathsf{AKE}}^{\text{ror-ake}}(\mathcal{A}_{\text{ror}}) \leq 2 \cdot \mathbf{Adv}_{\mathsf{AKE}}^{\text{ror-ake}}(t, q_{\text{send}}, q_{\text{reveal}} + 1, q_{\text{exe}})$. The lemma follows easily. ▌

Lemma 4.2 Let $\mathbf{Adv}_{\mathsf{AKE}}^{\text{ftg-ake}}(t, q_{\text{send}}, q_{\text{reveal}}, q_{\text{exe}})$ and $\mathbf{Adv}_{\mathsf{AKE}}^{\text{ror-ake}}(t, q_{\text{send}}, q_{\text{test}}, q_{\text{exe}})$ be defined as in Lemma 4.1. Then, for any protocol AKE, $\mathbf{Adv}_{\mathsf{AKE}}^{\text{ror-ake}}(t, q_{\text{send}}, q_{\text{test}}, q_{\text{exe}}) \leq q_{\text{test}} \cdot \mathbf{Adv}_{\mathsf{AKE}}^{\text{ftg-ake}}(t, q_{\text{send}}, q_{\text{reveal}}, q_{\text{exe}})$, where $q_{\text{reveal}} = q_{\text{test}} - 1$.

Proof: In order to prove this lemma, we show how to build a sequence of adversaries $\mathcal{A}_{\text{ftg}}^i$ against the indistinguishability of an authenticated key exchange protocol, AKE, in the FTG model given an adversary $\mathcal{A}_{\text{ror}}$ against the indistinguishability of the same protocol AKE in the ROR model. We know that $\mathcal{A}_{\text{ror}}$ has time-complexity at most t and that it asks at most q_{send}, q_{test}, and q_{exe} queries to its *Send*, *Test*, and *Execute* oracles, respectively.

The proof uses a standard hybrid argument, in which we define a sequence of $q_{\text{test}} + 1$ hybrid experiments V_i, where $0 \leq i \leq q_{\text{test}}$. In experiment V_i, the first i queries to the *Test* oracle are answered using a random key and all remaining *Test* queries are answered using the real key. Please note that the hybrid experiments at the extremes correspond to the real and random experiments in the definition of indistinguishability in the ROR model. Hence, in order to prove the bound in the lemma, it suffices to prove that the difference in probability that adversary $\mathcal{A}_{\text{ror}}$ returns 1 between any two consecutive experiments V_i and V_{i-1} is at most $\mathbf{Adv}_{\mathsf{AKE}}^{\text{ftg-ake}}(t, q_{\text{send}}, q_{\text{test}} - 1, q_{\text{exe}})$. This is achieved by building a sequence of q_{test} adversaries $\mathcal{A}_{\text{ftg}}^i$, as described below.

Let $\mathcal{A}_{\text{ftg}}^i$ be a distinguisher $\mathcal{A}_{\text{ftg}}^i$ for experiments V_i and V_{i-1}, where $1 \leq i \leq q_{\text{test}}$. $\mathcal{A}_{\text{ftg}}^i$ starts running $\mathcal{A}_{\text{ror}}$ answering to its queries as follows. If $\mathcal{A}_{\text{ror}}$ asks a *Send* or *Execute* query, then $\mathcal{A}_{\text{ftg}}$ answers it using its corresponding oracle. If $\mathcal{A}_{\text{ror}}$ asks a *Test* query, then $\mathcal{A}_{\text{ftg}}$ answers it with a random key if this query is among the first $i-1$. If this is the i-th *Test* query, then $\mathcal{A}_{\text{ftg}}$ uses its *Test* oracle to answer it. All remaining *Test* queries are answered using the output of the *Reveal* query. $\mathcal{A}_{\text{ftg}}$ finishes its execution by outputting the same guess bit b outputted by $\mathcal{A}_{\text{ror}}$.

Note that $\mathcal{A}_{\text{ftg}}^i$ has time-complexity at most t and asks at most q_{send}, $q_{\text{test}} - 1$, and q_{exe} queries to its *Send*, *Reveal*, and *Execute* oracles, respectively.

In order to analyze the advantage of $\mathcal{A}_{\text{ftg}}^i$, first notice that when *Test* oracle returns a random key, $\mathcal{A}_{\text{ftg}}^i$ runs $\mathcal{A}_{\text{ror}}$ exactly as in the experiment V_i. Next, notice that when *Test* oracle returns the real key, $\mathcal{A}_{\text{ftg}}^i$ runs $\mathcal{A}_{\text{ror}}$ exactly as in the experiment V_{i-1}. It follows that the difference in probability that adversary $\mathcal{A}_{\text{ror}}$ returns 1 between experiments V_i and V_{i-1} is at most $\mathbf{Adv}_{\mathsf{AKE}}^{\text{ftg-ake}}(\mathcal{A}_{\text{ror}}) \leq \mathbf{Adv}_{\mathsf{AKE}}^{\text{ftg-ake}}(t, q_{\text{send}}, q_{\text{test}} - 1, q_{\text{exe}})$. The lemma follows easily. ▌

Even though the reduction in Lemma 4.2 is not security-preserving (i.e., there is a non-constant factor loss in the reduction), it does not imply that a gap really exists— there might exist a tight reduction between the two notions that we have not yet found. In order to prove that the non-constant factor loss in the reduction is indeed intrinsic, we need to show that there exist schemes for which the gap does exist.

To achieve this goal, one can use techniques similar to those used to prove that a gap exists between the Left-Or-Right and Find-Then-Guess notions of security for symmetric encryption schemes [5]. In that paper, they show how to construct a new symmetric encryption scheme $\mathcal{E}'$ from a secure encryption scheme $\mathcal{E}$ such that $\mathcal{E}'$ exhibits the gap. $\mathcal{E}'$ was constructed in such a way that its encryption function works like the encryption function of $\mathcal{E}$ most of the time, except in a few cases (which are easily identifiable) in which the ciphertext it generates contains the plaintext. The probability in which such bad cases happen in their construction is exactly $1/q$, where q is the non-constant factor in the reduction. Using a similar technique, we can prove the following.

Proposition 4.3 *The gap exhibited in Lemma 4.2 is intrinsic and cannot be avoided.*

Proof Sketch: Let AKE be an authenticated key exchange protocol that is secure in the FTG model. For simplicity, let us assume that $q_{\text{test}} = 2^l$, for some integer l, and that k is the length of the session key. Then, using AKE, we can construct a new scheme AKE' that exhibits the claimed gap as follows. AKE' simply runs AKE and sets its session key sk to the one generated by AKE whenever any of the first l bits of the latter are different from 0. When the first l bits of the session key generated by AKE are to 0, then AKE' simply sets *all* the bits of the session key sk to 0.

First, let us analyze the ROR security of AKE'. To this end, consider an adversary $\mathcal{A}_{\text{ror}}$ against the ROR security of AKE', which makes $q_{\text{exe}} = 2^l$ queries to its *Execute* oracle and $q_{\text{test}} = 2^l$ queries to its *Test* oracle and then outputs 1 if any of the outputs of the *Test* oracle is a string containing all zeros and returns 0, otherwise. Let b be the hidden bit used to simulate *Test* oracle. To analyze the success probability of AKE', first notice that, if $b = 0$ (i.e., the *Test* oracle returns random keys), then the probability that the *Test* oracle returns the zero string is negligible if we assume that the session key length k is sufficiently larger than the parameter l. Thus, the probability that $\mathcal{A}_{\text{ror}}$ returns 0 in this case is also negligible. Second, notice that if $b = 1$ (i.e., the *Test* oracle returns the actual session keys), then the probability that the *Test* oracle returns the zero string is greater than 1/2. This is because the probability that the first l bits of each session key sk generated by AKE are equal to 0 is close to $1/2^l$ due to the security of the underlying protocol AKE. As a result, the probability that at least one of $q_{\text{test}} = 2^l$ session keys computed by AKE' is equal to zero string (in which case $\mathcal{A}_{\text{ror}}$ succeeds) is close to $1 - (1 - q_{\text{test}})^{q_{\text{test}}} \approx 1 - 1/e \geq 1/2$, where e is the base of the natural logarithm. Therefore, the ror-ake-advantage of $\mathcal{A}_{\text{ror}}$ and the ror-ake-advantage function of the protocol AKE' is at least 1/2.

Second, let us analyze the FTG security of AKE'. To do so, we note that an adversary against the FTG security of AKE' must hope that the first l bits of the challenge session key that it receives from its *Test* oracle are equal to zero. Otherwise, this adversary cannot achieve an advantage that is greater than that of an adversary attacking the underlying AKE. Since the first l bits of the challenge session key are equal to zero with probability close to $1/2^l$, it follows that $\mathbf{Adv}^{\text{ftg}-\text{ake}}_{\mathsf{AKE}'}(t, q_{\text{send}}, q_{\text{reveal}}, q_{\text{exe}}) \leq \mathbf{Adv}^{\text{ftg}-\text{ake}}_{\mathsf{AKE}}(t, q_{\text{send}}, q_{\text{reveal}}, q_{\text{exe}}) + 1/2^l$.

Finally, we observe that, since we started with a protocol AKE that is secure in the FTG sense, the value $\mathbf{Adv}^{\text{ftg}-\text{ake}}_{\mathsf{AKE}}(t, q_{\text{send}}, q_{\text{reveal}}, q_{\text{exe}})$ should be negligible in comparison to $1/2^l$. As a result, the ratio between the ror-ake-advantage and the ftg-ake-advantage of the protocol AKE' necessarily contains a non-constant factor that is proportional to the number of queries to the *Test* oracle. ▮

We note that, when the underlying scheme AKE is a password-based key exchange, not every choice of parameters yields the result claimed in the proposition due to the fact $\mathbf{Adv}_{\mathsf{AKE}}^{\mathrm{ftg-ake}}(t, q_{\mathrm{send}}, q_{\mathrm{reveal}}, q_{\mathrm{exe}})$ usually contains non-negligible terms. However, since there are choices of parameters and schemes for which the gap does exist, that is already sufficient for the purpose of the proposition.

Finally, it is also worth pointing out that the protocol AKE' used to exhibit the gap in Lemma 4.2 cannot be considered secure in the FTG model. This fact, however, does not invalidate the proof of Proposition 4.3.

5 A generic three-party password-based protocol

In this section, we introduce a generic construction of a 3-party password-based key exchange protocol in the scenario in which we have an *honest-but-curious* server. It combines a 2-party password-based key exchange, a secure key distribution protocol, and a 2-party MAC-based key exchange and has several attractive features. First, our construction does not rely on the Random Oracle (RO) model [8] as long as the underlying primitives themselves do not rely on it. Hence, by using schemes such as the KOY protocol [19] for the 2-party password-based key exchange and the 3-party key distribution scheme in [10], one gets a 3-party password-based protocol whose security is in the standard model. Second, if 2-party password-based key exchange protocols already exist between the server and its users in a distributed system, they can be re-used in the construction of our 3-party password-based key exchange.

5.1 Building blocks

In this section, we recall the definitions for the cryptographic primitives that we use in the construction of our generic 3-party password-based authenticated key exchange, GPAKE.

Decisional Diffie-Hellman assumption: DDH. The *decisional Diffie-Hellman* assumption, DDH, states, roughly, that the distributions (g^u, g^v, g^{uv}) and (g^u, g^v, g^w) are computationally indistinguishable when u, v, w are drawn at random from $\{1, \ldots, |\mathbb{G}|\}$. This can be made more precise by defining two experiments, $\mathbf{Exp}_{\mathbb{G}}^{\text{ddh-real}}(\mathcal{A})$ and $\mathbf{Exp}_{\mathbb{G}}^{\text{ddh-rand}}(\mathcal{A})$. In both experiments, we compute two values $U = g^u$ and $V = g^v$ to be given to $\mathcal{A}$. But in addition to that, we also provide a third input, which is g^{uv} in $\mathbf{Exp}_{\mathbb{G}}^{\text{ddh-real}}(\mathcal{A})$ and g^z for a random z in $\mathbf{Exp}_{\mathbb{G}}^{\text{ddh-rand}}(\mathcal{A})$. The goal of the adversary is to guess a bit indicating the experiment it thinks it is in. We define the **advantage** of $\mathcal{A}$ in violating the DDH assumption, $\mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}(\mathcal{A})$, as $\Pr[\,\mathbf{Exp}_{\mathbb{G}}^{\text{ddh-real}}(\mathcal{A}) = 1\,] - \Pr[\,\mathbf{Exp}_{\mathbb{G}}^{\text{ddh-rand}}(\mathcal{A}) = 1\,]$. The **advantage function** of the group, $\mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}(t)$ is then defined as the maximum value of $\mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}(\mathcal{A})$ over all $\mathcal{A}$ with time-complexity at most t.

Message authentication codes (MAC). Let ℓ be a security parameter and let sk be a ℓ-bit secret key uniformly distributed in $\{0,1\}^{\ell}$. A message authentication code $\mathsf{MAC} = (\mathsf{Tag}, \mathsf{Ver})$ is defined by the following two algorithms: (1) A *MAC generation algorithm* Tag, possibly probabilistic, which given a message m and a secret key $sk \in \{0,1\}^{\ell}$, produces a tag μ; and (2) A *MAC verification algorithm* Ver, which given a tag μ, a message m, and a secret key sk, outputs 1 if μ is a valid tag for m under sk and 0, otherwise.

The security notion that we need for the MAC scheme is strong existential unforgeability under chosen-message attacks, which is based on existential unforgeability notion in [6]. In this notion, the adversary should be unable to create a new valid message-tag pair, even after seeing many such valid pairs. More specifically, let MAC be a MAC scheme, let sk be a secret key chosen uniformly at random from $\{0,1\}^{\ell}$, and let $\mathcal{A}$ be an adversary against the

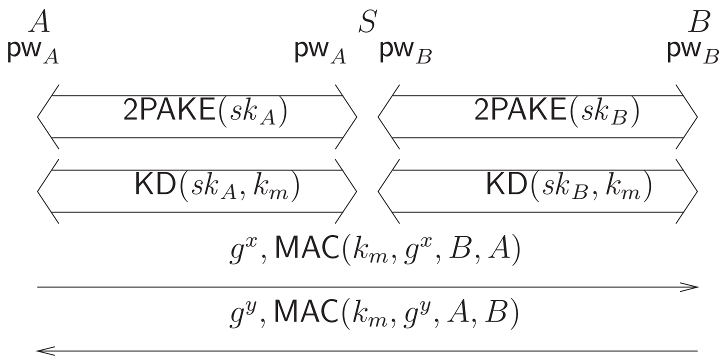

Figure 2: GPAKE: a generic three-party password-based key exchange

security of MAC. Then, consider the experiment in which the adversary $\mathcal{A}$ is given access to a MAC generation oracle $\mathsf{Tag}(sk;\cdot)$ and to a MAC verification oracle $\mathsf{Ver}(sk;\cdot,\cdot)$ and outputs a message-tag pair (m,μ). Let Succ denote the event in which $\mathsf{Ver}(sk;m,\mu)=1$ and the tag μ was not outputted by the $\mathsf{Tag}(sk;\cdot)$ oracle on input m. The **advantage** of $\mathcal{A}$ in violating the strong existential unforgeability of the MAC scheme MAC under chosen-message attacks is defined as $\mathbf{Adv}^{\mathrm{suf-cma}}_{\mathsf{MAC}}(\mathcal{A}) = \Pr[\,\textsc{Succ}\,]$. The **advantage function** of the MAC scheme MAC, $\mathbf{Adv}^{\mathrm{suf-cma}}_{\mathsf{MAC}}(t, q_g, q_v)$, is then defined as the maximum value of $\mathbf{Adv}^{\mathrm{suf-cma}}_{\mathsf{MAC}}(\mathcal{A})$ over all $\mathcal{A}$ with time-complexity at most t and asking at most q_g and q_v queries to its MAC generation and verification oracles, respectively.

3-party key distribution. A secure key distribution protocol KD is a 3-party protocol between 2 parties and a trusted server S where S picks a session key at random and securely sends it to the users. The security model, formally introduced in [10], is a generalization of that for 2-party authenticated key exchange protocols, to which a new oracle was added to represent the trusted server. Their security is in the Find-Then-Guess (FTG) model, using the terminology that we introduced for key exchange protocols.

In our generic construction, we only need a KD secure with respect to a single session since the symmetric keys used as input to the key distribution protocol differ from session to session. They are the session keys obtained from the 2-party password-based authenticated key exchange protocols between the server and each of the two parties. Since in this case, both the FTG and the ROR notions are equivalent, we opted to use their definition (i.e. FTG) adapted to our terminology. That is, we define $\mathbf{Adv}^{\mathrm{ftg-kd}}_{\mathsf{KD}}(\mathcal{A})$ as the **advantage** of adversary $\mathcal{A}$ in violating the indistinguishability of a key distribution KD in the FTG sense, and $\mathbf{Adv}^{\mathrm{ftg-kd}}_{\mathsf{KD}}(t,s,r)$ as the **advantage function** of KD, which is the maximum value of $\mathbf{Adv}^{\mathrm{ftg-kd}}_{\mathsf{KD}}(\mathcal{A})$ over all $\mathcal{A}$ with time-complexity at most t, asking *Send* queries with respect to at most s sessions and asking at most r *Reveal* queries.

5.2 Description of the generic solution

Our generic construction can be seen as a form of compiler transforming any secure 2-party password-based key exchange protocol P into a secure password-based 3-party key exchange protocol P' in the *honest-but-curious* security model using a secure key distribution KD, a secure MAC scheme, and generic number-theoretic operations in a group $\mathbb{G}$ for which the DDH assumption holds (see Section 5.1).

The compiler, depicted in Figure 2, works as follows. First, we use the protocol P between a user A and the server S to establish a secure high-entropy session key sk_A. Second, we use the protocol P between the server S and the user B in order to establish a session key sk_B. Third, using a key distribution KD, we have the server S first select a MAC key k_m, using the key generation of the latter, and then distribute this key to A and B using the session keys sk_A and sk_B, respectively, generated in the first two steps. Finally, A and B use a MAC-based key exchange to establish a session key CDH in an authenticated way.

5.3 Security

Indistinguishability in the Real-Or-Random model. As the following theorem states, the generic scheme GPAKE depicted in Figure 2 is a secure 3-party password-based key exchange protocol as long as the Decisional Diffie-Hellman assumption holds in $\mathbb{G}$ and the underlying primitives it uses are secure. The proof of security, however, is not as tight as one would wish, due to the loss of a factor 2 in the reduction.

Theorem 5.1 Let $\mathsf{2PAKE}$ be a secure 2-party password-based Key Exchange, KD be a secure key distribution, and MAC be a secure MAC algorithm. Let $q_{\rm exe}$ and $q_{\rm test}$ represent the number of queries to *Execute* and *Test* oracles, and let $q^A_{\rm send}$, $q^B_{\rm send}$, $q_{\rm kd}$, and $q_{\rm ake}$ represent the number of queries to the *SendClient* and *SendServer* oracles with respect to each of the two $\mathsf{2PAKE}$ protocols, the KD protocol, and the final AKE protocol. Then,

$$\begin{aligned}
&\mathbf{Adv}^{\rm ror-ake}_{\mathsf{GPAKE},\mathcal{D}}(t, q_{\rm exe}, q_{\rm test}, q^A_{\rm send}, q^B_{\rm send}, q_{\rm kd}, q_{\rm ake}) \leq \\
&\quad 2\cdot \mathbf{Adv}^{\rm ror-ake}_{\mathsf{2PAKE},\mathcal{D}}(t, q_{\rm exe}, q_{\rm exe}+q^A_{\rm send}, q^A_{\rm send}) + \ 2\cdot \mathbf{Adv}^{\rm ror-ake}_{\mathsf{2PAKE},\mathcal{D}}(t, q_{\rm exe}, q_{\rm exe}+q^B_{\rm send}, q^B_{\rm send}) \\
&\quad + 2\cdot (q_{\rm exe}+q_{\rm kd})\cdot \mathbf{Adv}^{\rm ftg-kd}_{\mathsf{KD}}(t,1,0) + \ 2\cdot q_{\rm ake}\cdot \mathbf{Adv}^{\mathsf{euf-cma}}_{\mathsf{MAC}}(t,2,0) \\
&\quad + 2\cdot \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t \ + \ 8(q_{\rm exe}+q_{\rm ake})\tau_e)\ ,
\end{aligned}$$

where τ_e denotes the exponentiation computational time in $\mathbb{G}$.

Key privacy with respect to the server. As the following theorem states, the generic scheme GPAKE depicted in Figure 2 has key privacy with respect to the server as long as the Decisional Diffie-Hellman assumption holds in $\mathbb{G}$.

Theorem 5.2 Let GPAKE be the 3-party password-based authenticated key exchange scheme depicted in Figure 2. Then,

$$\mathbf{Adv}^{\rm kp-ake}_{\mathsf{GPAKE},\mathcal{D}}(t, q_{\rm exe}, q_{\rm test}, q^A_{\rm send}, q^B_{\rm send}, q_{\rm kd}, q_{\rm ake}) \quad \leq \quad 2\cdot \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t \ + \ 8(q_{\rm exe}+q_{\rm ake})\tau_e)\ ,$$

where the parameters are defined as in Theorem 5.1.

5.4 Instantiations

Several practical schemes can be used in the instantiation of the 2-party password-based key exchange of our generic construction. Among them are the KOY protocol [19] and its generalization [17], the PAK suite [22], and several other schemes based on the encrypted key exchange scheme of Bellovin and Merritt [11] (e.g., [14]).

In the instantiation of the key distribution scheme, one could use the original proposal in [10] or any other secure key distribution scheme. In particular, the server could use a chosen-ciphertext secure symmetric encryption scheme to distribute the keys to the users. Independently of the choice, one should keep in mind that the security requirements for the key distribution scheme are very weak. It only needs to provide security with respect to one session.

For the instantiation of the MAC, any particular choice that makes the MAC term in Theorem 5.1 negligible will do. Possible choices are the HMAC [4] or the CBC MAC.

It is important to notice that, in order for GPAKE to be secure, the underlying 2-party password-based protocol *must* be secure in the ROR model. In view of the computational gap that exists between the ROR and the FTG models (see Proposition 4.3), a 2-party password-based secure in the FTG model does not suffice to prove the security of GPAKE.

6 Proof of security for GPAKE

Indistinguishability of GPAKE in the ROR model. For simplicity, we assume the set of honest users contains only users A and B. The solution can be easily extended to the more general case with essentially the same bounds.

Let $\mathcal{A}$ be an adversary against the indistinguishability of GPAKE in the Real-Or-Random sense with time-complexity at most t, and asking at most $q_{\rm exe}$ queries to its *Execute* oracle, $q_{\rm test}$ queries to its *Test* oracle, $q_{\rm send}^{A}$ queries to *SendClient* and *SendServer* oracles with respect to the 2PAKE protocol between A and the trusted server S, $q_{\rm send}^{B}$ queries with respect to the 2PAKE protocol between B and S, $q_{\rm kd}$ queries with respect to the KD protocol, and $q_{\rm ake}$ queries with respect to the final authenticated key exchange protocol.

Our proof consists of a sequence of hybrid experiments, starting with the real attack against GPAKE scheme and ending in an experiment in which the adversary's advantage is 0, and for which we can bound the difference in the adversary's advantage between any two consecutive experiments. For each experiment $\mathbf{Exp}_n$, we define an event $\textsc{Succ}_n$ corresponding to the case in which the adversary correctly guesses the hidden bit b involved in the *Test* queries (see Section 3).

Experiment $\mathbf{Exp}_0$. This experiment corresponds to the real attack. By definition, we have

$$\mathbf{Adv}^{\rm ror-ake}_{\mathsf{GPAKE},\mathcal{D}}(\mathcal{A}) \quad = \quad 2\cdot \Pr[\,\textsc{Succ}_0\,] - 1 \tag{1}$$

Experiment $\mathbf{Exp}_1$. We now modify the simulation of the oracles as follows. We replace the session key sk_A used as input to the KD protocol by a random session key sk'_A in all of the sessions. As the following lemma shows, the difference between the success probability of the adversary $\mathcal{A}$ between the current and previous experiments is at most twice the probability of breaking the security of the underlying 2PAKE protocol between A and S.

Lemma 6.1 $\left|\Pr[\,\textsc{Succ}_1\,] - \Pr[\,\textsc{Succ}_0\,]\right| \le \mathbf{Adv}^{\rm ror-ake}_{\mathsf{2PAKE},\mathcal{D}}(t, q_{\rm exe}, q_{\rm exe} + q^{A}_{\rm send}, q^{A}_{\rm send})$.

Proof: Let $\mathcal{A}_1$ be a distinguisher for experiments $\mathbf{Exp}_1$ and $\mathbf{Exp}_0$. We can build an adversary $\mathcal{A}_{\sf pake}$ against the indistinguishability of the 2PAKE protocol between A and S using $\mathcal{A}_1$ as follows. $\mathcal{A}_{\sf pake}$ starts by choosing a bit b uniformly at random and selecting the passwords for all users in the system except A according to the distribution of $\mathcal{D}$. Next, it starts running $\mathcal{A}_1$, giving it the passwords for all the malicious clients in the system, and answering to its oracle queries as follows.

- *SendClient* and *SendServer* queries. If $\mathcal{A}_1$ asks a *SendClient* or *SendServer* query pertaining to an instance of the 2PAKE protocol between B and S, then $\mathcal{A}_{\sf pake}$ can answer it using the password of client B that it has picked at the beginning of its execution. If the

SendClient or *SendServer* query pertains to an instance of the 2PAKE protocol between A and S, then $\mathcal{A}_{\mathsf{pake}}$ can answer it by asking the corresponding query to its *Send* oracle. If this query forces the given instance of client A or S to accept, then we also ask a *Test* query to that instance (unless *Test* query had already been asked to its partner). The output of this *Test* query will be used as the session key shared between A and S. All remaining *SendClient* and *SendServer* queries by $\mathcal{A}_1$ can be easily answered either using the chosen values for the session key shared between A and S or the session keys computed in the execution of the 2PAKE protocol between S and B.

- *Execute* queries. $\mathcal{A}_{\mathsf{pake}}$ can easily answer these queries using its own *Execute* oracle and the output of the *Test* queries, as in the simulation of *SendClient* and *SendServer* queries.

- *Test* queries. $\mathcal{A}_{\mathsf{pake}}$ can easily answer these queries using the bit b that it has previously selected and the session keys that it has computed.

Let b' be the output of $\mathcal{A}_1$. If $b' = b$, then $\mathcal{A}_{\mathsf{pake}}$ outputs 1. Otherwise, it outputs 0.

One can easily see that the probability that $\mathcal{A}_{\mathsf{pake}}$ outputs 1 when its *Test* oracle returns real keys is exactly the probability that $\mathcal{A}_1$ correctly guesses the hidden bit b in experiment $\mathbf{Exp}_0$ (i.e., $\Pr[\,\textsc{Succ}_0\,]$). Similarly, the probability that $\mathcal{A}_{\mathsf{pake}}$ outputs 1 when its *Test* oracle returns random keys is exactly the probability that $\mathcal{A}_1$ correctly guesses the hidden bit b in experiment $\mathbf{Exp}_1$ (i.e., $\Pr[\,\textsc{Succ}_1\,]$). The lemma follows by noticing that $\mathcal{A}_{\mathsf{pake}}$ has at most time-complexity t and asks at most $q_{\mathrm{exe}} + q^{A}_{\mathrm{send}}$ queries to its *Test* oracle, at most q_{exe} queries to its *Execute* oracle, and at most q^{A}_{send} queries to its *Send* oracle. ▌

Experiment $\mathbf{Exp}_2$. We modify the previous experiment by replacing the session key sk_B used as input to the KD protocol by a random session key sk'_B in all of the sessions. Using similar arguments, one can prove the following lemma.

Lemma 6.2 $\left|\Pr[\,\textsc{Succ}_2\,] - \Pr[\,\textsc{Succ}_1\,]\right| \le \mathbf{Adv}^{\mathrm{ror-ake}}_{\mathsf{2PAKE},\mathcal{D}}(t, q_{\mathrm{exe}}, q_{\mathrm{exe}} + q^{B}_{\mathrm{send}}, q^{B}_{\mathrm{send}})$.

Experiment $\mathbf{Exp}_3$. In this experiment, we replace the MAC key k_m obtained via the key distribution protocol with a random key in all of the sessions involving both A and B. According to the following lemma, the difference between the success probability of the adversary $\mathcal{A}$ between the current and previous experiments is at most that of breaking the security of the key distribution scheme KD protocol among A, B, and S.

Lemma 6.3 $\left|\Pr[\,\textsc{Succ}_3\,] - \Pr[\,\textsc{Succ}_2\,]\right| \le (q_{\mathrm{exe}} + q_{\mathrm{kd}})\ \mathbf{Adv}^{\mathrm{ftg-kd}}_{\mathsf{KD}}(t, 1, 0)$.

Proof: The proof of of this lemma uses a sequence of hybrid experiments V_j, where j is an index between 0 and $q_s = (q_{\mathrm{exe}} + q_{\mathrm{kd}})$. Let i represent the i-th session involving honest users A and B. We define the hybrid experiment V_j as follows. If $i \le j$, then the MAC key k_m is chosen uniformly at random as in experiment $\mathbf{Exp}_3$. If $i > j$, then the MAC key k_m is computed via the key distribution protocol as in experiment $\mathbf{Exp}_2$. Note that experiments V_0 and V_{q_s} are equivalent to experiments $\mathbf{Exp}_2$ and $\mathbf{Exp}_3$, respectively. Let P_j be the probability of the event $\textsc{Succ}$ in Experiment V_j. Since $P_0 = \Pr[\,\textsc{Succ}_2\,]$ and $P_{q_s} = \Pr[\,\textsc{Succ}_3\,]$, it follows that $\left|\Pr[\,\textsc{Succ}_3\,] - \Pr[\,\textsc{Succ}_2\,]\right| = \sum_{j=1}^{q_s} \left|P_j - P_{j-1}\right|$. Thus, to prove the present lemma, it suffices to show that $\left|P_j - P_{j-1}\right|$ is at most $\mathbf{Adv}^{\mathrm{ftg-kd}}_{\mathsf{KD}}(t, 1, 0)$. To do so, we assume the existence of a distinguisher $\mathcal{A}_{3,j}$ for experiments V_{j-1} and V_j and we show how to build an adversary $\mathcal{A}^{j}_{\mathsf{kd}}$

against the KD protocol that has a success probability similar to that of $\mathcal{A}_{3,i}$ and that asks queries to its *Send* oracle with respect to a *single* session only.

The description of $\mathcal{A}_{\mathsf{kd}}^{j}$ is as follows. As in previous cases, $\mathcal{A}_{\mathsf{kd}}^{j}$ starts by choosing a bit b uniformly at random and selecting the passwords for all users in the system according to the distribution of $\mathcal{D}$. Next, $\mathcal{A}_{\mathsf{kd}}^{j}$ starts running $\mathcal{A}_{3,j}$, giving it the passwords for all the malicious clients in the system. Then, for the first $j-1$ sessions, $\mathcal{A}_{\mathsf{kd}}^{j}$ simulates all oracles exactly as in experiment $\mathbf{Exp}_3$. Likewise, for the last $q_s - j$ sessions, $\mathcal{A}_{\mathsf{kd}}^{j}$ simulates all oracles exactly as in experiment $\mathbf{Exp}_2$. For the j-th session, instead of choosing the inputs for the key distribution protocol, $\mathcal{A}_{\mathsf{kd}}^{j}$ simulates the *SendClient*, *SendServer*, and *Execute* oracles of $\mathcal{A}_{3,j}$ using the *Send* oracle for the KD protocol. Moreover, $\mathcal{A}_{\mathsf{kd}}^{j}$ also makes a *Test* query with respect to this session to obtain a key $\tilde{k_m}$ and it uses this key to simulate the remainder of the GPAKE protocol of the j-th session. Let b' be the output of $\mathcal{A}_{3,j}$. If $b' = b$, then $\mathcal{A}_{\mathsf{kd}}^{j}$ outputs 1. Otherwise, it outputs 0.

In order to analyze the advantage of $\mathcal{A}_{\mathsf{kd}}^{i}$, one can easily see that the probability that $\mathcal{A}_{\mathsf{kd}}^{i}$ outputs 1 when its *Test* oracle returns real keys is exactly the probability that $\mathcal{A}_{3,j}$ correctly guesses the hidden bit b in experiment V_{j-1} (i.e., P_{j-1}). Similarly, the probability that $\mathcal{A}_{\mathsf{kd}}^{i}$ outputs 1 when its *Test* oracle returns random keys is exactly the probability that $\mathcal{A}_{3,j}$ correctly guesses the hidden bit b in experiment V_j (i.e., P_{j-1}). The lemma follows by noticing that $\mathcal{A}_{\mathsf{kd}}^{i}$ has time-complexity at most t, that it asks queries to its *Send* oracle with respect to a *single* session only, and that it asks no *Reveal* queries. ▮

Experiment $\mathbf{Exp}_4$. In this experiment, we modify the oracle instances as follows. If the adversary asks a *SendClient* query containing a new pair message-tag not previously generated by an oracle, then we consider the MAC tag invalid and force the instance in question (which received a forged message) to terminate without accepting. As the following lemma shows, the difference between the current and previous experiments should be negligible if we use a secure MAC scheme.

Lemma 6.4 $\left|\Pr[\,\textsc{Succ}_4\,] - \Pr[\,\textsc{Succ}_3\,]\right| \leq q_{\text{ake}} \cdot \mathbf{Adv}_{\mathsf{MAC}}^{\text{suf-cma}}(t, 2, 0)$.

Proof: The proof of this lemma also uses hybrid arguments in the same way as in the proof of Lemma 6.3. The total number of hybrids in this case is q_{ake}, since *Execute* queries do not need to be taken into account in this case. In hybrid V_i, where $0 \leq i \leq q_{\text{ake}}$, queries in the first i sessions are answered as in experiment $\mathbf{Exp}_4$ and all other queries are answered as in experiment $\mathbf{Exp}_3$. Let $\mathcal{A}_{4,i}$ be a distinguisher for hybrids V_i and V_{i-1}. Using $\mathcal{A}_{4,i}$, we can build an adversary for the MAC scheme as follows.

For the first $i-1$ sessions, the adversary $\mathcal{A}_{\mathsf{mac}}^{i}$ will choose random values for the MAC key and is therefore able to perfect simulate the oracles given to $\mathcal{A}_{4,i}$. In the i-th session, $\mathcal{A}_{\mathsf{mac}}^{i}$ makes use of its MAC generation and verification oracles to answer queries from $\mathcal{A}_{4,i}$. If $\mathcal{A}_{4,i}$ asks a *SendClient* containing a pair message-tag not previously generated by $\mathcal{A}_{\mathsf{mac}}^{i}$, then $\mathcal{A}_{\mathsf{mac}}^{i}$ halts and outputs that pair as its forgery. If no such pair is generated, we output a failure indication. For all remaining sessions, $\mathcal{A}_{\mathsf{mac}}^{i}$ uses the actual MAC keys obtained via the key distribution scheme as in experiment $\mathbf{Exp}_3$ to answer queries from $\mathcal{A}_{4,i}$.

Let F be the event in which a message-tag pair is considered invalid in hybrid V_i but valid in hybrid V_{i-1}. Notice that $\Pr[F]$ is at most the probability that an adversary $\mathcal{A}_{\mathsf{mac}}^{i}$ can forge a new message-tag pair under a chosen-message attack. Since $\mathcal{A}_{\mathsf{mac}}^{i}$ has time-complexity t and makes at most two queries to its MAC generation oracle (to answer the *SendClient* queries) and no queries to its verification oracle, we have that $\Pr[F] \leq \mathbf{Adv}_{\mathsf{MAC}}^{\text{suf-cma}}(t, 2, 0)$. Moreover, since the two hybrids proceed identically until F occurs, we have $\Pr[\textsc{Succ}_{V_{i-1}} \wedge \neg F] = \Pr[\textsc{Succ}_{V_i} \wedge \neg F]$. By Lemma 1 of [25], we have $|\Pr[\textsc{Succ}_{V_{i-1}}] - \Pr[\textsc{Succ}_{V_i}]| \leq \Pr[F]$. The lemma follows from the fact that there are at most q_{ake} hybrids. ▌

Experiment $\mathbf{Exp}_5$. In this experiment, we change the simulation of *SendClient*(U^i, m) queries pertaining the last two flows of GPAKE as well as the simulation of the *Test*(U^i) oracle to avoid relying on the knowledge of the values x and y used to compute the answer to these queries. More precisely, instead of choosing the values of x and y directly, we assume that we are given a random DDH triple (X, Y, Z), where $X = g^x$, $Y = g^y$, and $Z = g^{xy}$. Then, using (X, Y, Z) and the classical random self-reducibility of the Diffie-Hellman problem, we show how to simulate the above-mentioned oracles for all those sessions involving two honest users. All other queries are simulated as in experiment $\mathbf{Exp}_4$.

The behavior of our simulation in this experiment is as follows. Experiment $\mathbf{Exp}_5$ is identical to experiment $\mathbf{Exp}_4$, except that we apply the following special rules when dealing with *Test*(U^i) and *SendClient*(U^i, m) queries for the last two flows of GPAKE where U^i and its intended partner are honest users:

R1: When processing a *SendClient*$(A^i, Start)$ query, the simulator selects two random values a_0 and x_0 in $\mathbb{Z}q$, computes $X_0 = X^{a_0} g^{x_0}$, and stores (a_0, x_0, X_0) in a table $\mathcal{X}$.

R2: When processing a *SendClient*$(B^j, (X_0, m_0))$ query in the last message of the protocol,

- if the element X_0 has been computed by our simulator and is thus stored in the table $\mathcal{X}$, then our simulator answers this query by choosing two random values $b_0, y_0 \overset{R}{\leftarrow} \mathbb{Z}q$ and computing $Y_0 = Y^{b_0} g^{y_0}$. It also stores (b_0, y_0, Y_0) in a table $\mathcal{Y}$. and computes the value $Z_0 = Z^{a_0 b_0} \times Y^{x_0 b_0} \times X^{a_0 y_0} \times g^{x_0 y_0}$ to be able to answer *Test* queries.
- if the elements X_0 is not stored in the table $\mathcal{X}$, then the simulation proceeds as in experiment $\mathbf{Exp}_4$ (i.e., it halts without accepting).

R3: When processing a *Test*(U^i) query, where U^i and its intended partner are instances of a honest user that have accepted, then the simulator answers this query using the value Z_0 that it has pre-computed. We notice that, since active attacks against such instances have been ruled out in experiment $\mathbf{Exp}_4$, the simulator always knows the correct value Z_0 for the session key.

In order to analyze the differences between the current and the previous experiments, we first observe that, since MAC forgeries have been dealt with in experiment $\mathbf{Exp}_4$, the second case of rule **R2** will always cause the user instance involved in that query to not accept. Second, for all those sessions involving instances of users A and B, the *Test*(U^i) queries are always correctly answered. As a result, it is clear that experiments $\mathbf{Exp}_4$ and $\mathbf{Exp}_5$ are equivalent, since we have consistently replaced one set of random variables by another set of identically distributed random variables. In particular, $\Pr[\textsc{Succ}_4] = \Pr[\textsc{Succ}_5]$.

Experiment $\mathbf{Exp}_6$. The current experiment is similar to the previous one, except that all rules are computed using a triple (X, Y, Z) sampled from a random distribution (g^x, g^y, g^z), intead of a DDH triple. As the following lemma shows, the difference between the current and previous experiments should be negligible if DDH is hard in $\mathbb{G}$.

Lemma 6.5 $\left|\Pr[\,\textsc{Succ}_6\,] - \Pr[\,\textsc{Succ}_5\,]\right| \leq \mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}(t \;+\; 8(q_{\text{exe}} + q_{\text{ake}})\tau_e)$.

Proof: Let $\mathcal{A}$ be an attacker that breaks the indistinguishability experiment of GPAKE with a different advantage in Experiment $\mathbf{Exp}_6$ than in Experiment $\mathbf{Exp}_5$. We can build an adversary $\mathcal{A}_{\mathsf{ddh}}$ for the DDH problem in $\mathbb{G}$ as follows. Let (X, Y, Z) be the input given to $\mathcal{A}_{\mathsf{ddh}}$. $\mathcal{A}_{\mathsf{ddh}}$ first selects a bit b at random and then starts running $\mathcal{A}$. If $\mathcal{A}$ asks a *SendClient*, *Execute*, or *Test* query, then $\mathcal{A}_{\mathsf{ddh}}$ computes its output exactly as in the previous experiment but using the triple (X, Y, Z) that it had received as input. Let b' be the output of $\mathcal{A}$. If $b' = b$, then $\mathcal{A}_{\mathsf{ddh}}$ returns 1 or 0, otherwise.

Let us now analyze the success probability of $\mathcal{A}_{\mathsf{ddh}}$. Clearly, when the triple (X, Y, Z) is a true Diffie-Hellman triple, $\mathcal{A}_{\mathsf{ddh}}$ runs $\mathcal{A}$ exactly as in experiment $\mathbf{Exp}_5$ and thus the probability that $\mathcal{A}_{\mathsf{ddh}}$ outputs 1 is exactly $\Pr[\text{Succ}_5]$. On the other hand, when (X, Y, Z) is a random triple, $\mathcal{A}_{\mathsf{ddh}}$ runs $\mathcal{A}$ exactly as in experiment $\mathbf{Exp}_6$ and thus the probability that $\mathcal{A}_{\mathsf{ddh}}$ outputs 1 is exactly $\Pr[\text{Succ}_6]$. The lemma follows from the fact that $\mathcal{A}_{\mathsf{ddh}}$ has time-complexity at most $t + 8(q_{\text{exe}} + q_{\text{ake}})\tau_e$, due to the additional time for the computations of the random self-reducibility. ▌

Due to the random self-reducibility property of the Diffie-Hellman problem, all the sessions keys Z_0 used to answer *Test* queries in experiment $\mathbf{Exp}_6$ are randomly and independently distributed in $\mathbb{G}$. As a result, no information on the hidden bit b used by the *Test* oracle is leaked to the adversary and thus $\Pr[\text{Succ}_6] = \frac{1}{2}$. This result combined with the previous lemmas yields the result in Theorem 5.1. ▌

Key privacy. The proof of key privacy uses arguments similar to those used in experiments $\mathbf{Exp}_5$ and $\mathbf{Exp}_6$ in the proof of indistinguishability of GPAKE. Let $\mathcal{A}_{\text{kp}}$ be an adversary against the key privacy of GPAKE with time-complexity at most t, and asking at most q_{exe} queries to its *Execute* oracle, q_{test} queries to its *TestPair* oracle, and q_{ake} queries to *SendClient* oracle with respect to the final MAC-based authenticated key exchange protocol. Using $\mathcal{A}_{\text{kp}}$, we can build an adversary $\mathcal{A}_{\mathsf{ddh}}$ for the DDH problem in $\mathbb{G}$ as follows.

Let (X, Y, Z) be the input given to $\mathcal{A}_{\mathsf{ddh}}$. $\mathcal{A}_{\mathsf{ddh}}$ first chooses the passwords for all users in the system according to the distribution of $\mathcal{D}$. It also chooses a bit b at random that is used to answer queries to the *TestPair* oracle. It then starts running $\mathcal{A}_{\text{kp}}$ giving all the password of all users to it. Since $\mathcal{A}_{\mathsf{ddh}}$ knows the password of all users, it can easily answer queries made by $\mathcal{A}_{\text{kp}}$. However, in order to use $\mathcal{A}_{\text{kp}}$ to help it solve the DDH problem, $\mathcal{A}_{\mathsf{ddh}}$ will use the classical random self-reducibility of the Diffie-Hellman problem to introduce its input triple in the answers to *SendClient*, *Execute*, and *TestPair* queries with respect to the last two flows of GPAKE.

To simulate the *Execute* oracle, we simply use the passwords that we have chosen and proceed as in the actual protocol, except when computing the last two flows of GPAKE. For these flows, we proceed as in simulation of the *SendClient* oracle described below.

The simulation of the *SendClient* and *TestPair* are as follows:

R1: When processing a *SendClient*(A^i, Start) query, $\mathcal{A}_{\mathsf{ddh}}$ selects two random values a_0 and x_0 in $\mathbb{Z}q$, computes $X_0 = X^{a_0} g^{x_0}$, and stores (a_0, x_0, X_0) in a table $\mathcal{X}$.

R2: When processing a *SendClient*$(B^j, (X_0, m_0))$ query,

- if the element X_0 has been computed by $\mathcal{A}_{\mathsf{ddh}}$ and is thus stored in the table $\mathcal{X}$, then $\mathcal{A}_{\mathsf{ddh}}$ answers this query by choosing two random values $b_0, y_0 \overset{R}{\leftarrow} \mathbb{Z}q$ and computing $Y_0 = Y^{b_0} g^{y_0}$. $\mathcal{A}_{\mathsf{ddh}}$ also stores (b_0, y_0, Y_0) in a table $\mathcal{Y}$ and computes $Z_0 = Z^{a_0 b_0} \times Y^{x_0 b_0} \times X^{a_0 y_0} \times g^{x_0 y_0}$ to be able to answer *TestPair* queries.

- if the elements X_0 is not stored in the table $\mathcal{X}$, then $\mathcal{A}_{\mathsf{ddh}}$ proceeds with the simulation as it would in a real attack.

R3: When processing a *TestPair*(U_1^i, U_2^j) query, $\mathcal{A}_{\mathsf{ddh}}$ first checks whether U_1^i and U_2^j have both accepted and have the same key. If the check fails, then $\mathcal{A}_{\mathsf{ddh}}$ returns $\perp$. If the check passes, then $\mathcal{A}_{\mathsf{ddh}}$ knows the corresponding value Z_0 for the secret key and can answer it based on the hidden bit b it had previously chosen.

Let b' be the output of $\mathcal{A}_{\text{kp}}$. If $b' = b$, then $\mathcal{A}_{\mathsf{ddh}}$ returns 1 and 0, otherwise.

We would like to observe here that the second case of rule **R2** has no influence over *TestPair* queries, since the latter can only be asked to pair of oracle instances which share the same key. This is because even though the instance involved in the *SendClient* may itself accept, its partner would not be an oracle instance. Hence, a *TestPair* query involving this instance would always return the invalid symbol $\perp$.

In order to analyze the success probability of $\mathcal{A}_{\mathsf{ddh}}$, first consider the case in which the triple (X, Y, Z) is a true Diffie-Hellman triple. Then, in this case, one can see that simulation of the $\mathcal{A}_{\text{kp}}$ oracles is perfect. Hence, the probability that $\mathcal{A}_{\mathsf{ddh}}$ outputs 1 is exactly $\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{kp-ake}}_{\mathsf{GPAKE},\mathcal{D}}(\mathcal{A}_{\text{kp}})$. On the other hand, when (X, Y, Z) is a random triple, the keys Z_0 used to answer *TestPair* queries are all random and independent as a result of the random self-reducibility property of the Diffie-Hellman problem. Hence, no information on b is leaked through *TestPair* queries and the probability that $\mathcal{A}_{\mathsf{ddh}}$ outputs 1 is exactly $\frac{1}{2}$ in this case. The proof of Theorem 5.2 follows from the fact that $\mathcal{A}_{\mathsf{ddh}}$ has time-complexity at most $t + 8(q_{\text{exe}} + q_{\text{ake}})\tau_e$, due to the additional time for the computations of the random self-reducibility.

7 Concluding remarks

Authentication. In order to take explicit authentication into account, one can easily extend our model using definitions similar to those of Bellare *et al.* [7] for unilateral or mutual authentication. In their definition, an adversary is said to break authentication if it succeeds in making any oracle instance terminate the protocol without a partner oracle. Likewise, one could also use their generic transformation to enhance our generic construction so that it provides unilateral or mutual authentication. The drawback of using their generic transformation is that it requires the random oracle model. However, standard solutions based on pseudorandom function families that do not rely on the random oracle model can also be used in this case.

More efficient constructions. Even though the generic construction presented in this paper is quite practical, more efficient solutions are possible. For instance, one possible modification is to replace the key distribution phase of our generic construction by a single round of communication in which the server encrypts the MAC key using authenticated encryption scheme. Another possible improvement to our generic construction would be to combine the key distribution and the final key exchange phases into a single phase. One can easily think of different solutions for this scenario that are more efficient that the one we give. Nevertheless, the overall gain in efficiency that could be obtained with these extensions would most likely not be very significant since the most costly part of these two phases, the Diffie-Hellman protocol, seems to be necessary if key privacy with respect to the server is to be achieved. As a result, we believe that the best way to improve the efficiency of generic construction is to adapt specific solutions in the 2-party model to the 3-party model, instead of treating these schemes as black boxes.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. The work described in this document has been supported in part by France Telecom R&D under the contract CIDRE, between France Telecom R&D and École normale supérieure, and in part by the European Commission through the IST Program under the contract IST-2002-507932 ECRYPT.

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158, San Francisco, CA, USA, Apr. 8–12, 2001. Springer-Verlag, Berlin, Germany. (Cited on page 7.)

[2] M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In S. Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84, Les Diablerets, Switzerland, Jan. 23–26, 2005. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[3] Advanced encryption standard (aes). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, Nov. 2001. (Cited on page 3.)

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, CA, USA, Aug. 18–22, 1996. Springer-Verlag, Berlin, Germany. (Cited on page 17.)

[5] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, Oct. 19–22, 1997. IEEE Computer Society Press. (Cited on page 5, 13.)

[6] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000. (Cited on page 14.)

[7] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer-Verlag, Berlin, Germany. (Cited on page 4, 5, 6, 8, 9, 22.)

[8] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, Nov. 3–5, 1993. ACM Press. (Cited on page 4, 14.)

[9] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, CA, USA, Aug. 22–26, 1994. Springer-Verlag, Berlin, Germany. (Cited on page 4, 9.)

[10] M. Bellare and P. Rogaway. Provably secure session key distribution — the three party case. In *28th Annual ACM Symposium on Theory of Computing*, pages 57–66, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press. (Cited on page 4, 8, 9, 14, 15, 16.)

[11] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84, Oakland, California, USA, May 1992. IEEE Computer Society Press. (Cited on page 2, 4, 8, 16.)

[12] M. K. Boyarsky. Public-key cryptography and password protocols: The multi-user case. In *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 63–72, Kent Ridge Digital Labs, Singapore, Nov. 1–4, 1999. ACM Press. (Cited on page 3, 8.)

[13] V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171, Bruges, Belgium, May 14–18, 2000. Springer-Verlag, Berlin, Germany. (Cited on page 4.)

[14] E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. In F. Bao, R. Deng, and J. Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 145–158, Singapore, Mar. 1–4, 2004. Springer-Verlag, Berlin, Germany. (Cited on page 3, 8, 16.)

[15] J. W. Byun, I. R. Jeong, D. H. Lee, and C.-S. Park. Password-authenticated key exchange between clients with different passwords. In R. H. Deng, S. Qing, F. Bao, and J. Zhou, editors, *ICICS 02: 4th International Conference on Information and Communication Security*, volume 2513 of *Lecture Notes in Computer Science*, pages 134–146, Singapore, Dec. 9–12, 2002. Springer-Verlag, Berlin, Germany. (Cited on page 4.)

[16] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on page 4.)

[17] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany. `http://eprint.iacr.org/2003/032.ps.gz`. (Cited on page 3, 4, 8, 16.)

[18] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 408–432, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer-Verlag, Berlin, Germany. `http://eprint.iacr.org/2000/057`. (Cited on page 4.)

[19] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag, Berlin, Germany. (Cited on page 3, 4, 8, 14, 16.)

[20] H. Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[21] C.-L. Lin, H.-M. Sun, and T. Hwang. Three-party encrypted key exchange: Attacks and a solution. *ACM SIGOPS Operating Systems Review*, 34(4):12–20, Oct. 2000. (Cited on page 4.)

[22] P. D. MacKenzie. The PAK suite: Protocols for password-authenticated key exchange. Contributions to IEEE P1363.2, 2002. (Cited on page 3, 8, 16.)

[23] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the Association for Computing Machinery*, 21(21):993–999, Dec. 1978. (Cited on page 2, 4.)

[24] V. Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. (Cited on page 8.)

[25] V. Shoup. OAEP reconsidered. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 239–259, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer-Verlag, Berlin, Germany. (Cited on page 20.)

[26] J. G. Steiner, B. C. Neuman, and J. L. Schiller. Kerberos: An authentication service for open networks. In *Proceedings of the USENIX Winter Conference*, pages 191–202, Dallas, TX, USA, 1988. (Cited on page 1, 4.)

[27] M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. *ACM SIGOPS Operating Systems Review*, 29(3):22–30, July 1995. (Cited on page 4.)