

# Password Based Server Aided Key Exchange<sup>\*†</sup>

Yvonne Cliff            Yiu Shing Terry Tin  
Colin Boyd

Information Security Institute, Queensland University of Technology  
GPO Box 2434, Brisbane Q 4001, Australia.  
y.cliff@isi.qut.edu.au, {t.tin,c.boyd}@qut.edu.au

April 21, 2006

## Abstract

We propose a new password-based 3-party protocol with a formal security proof in the standard model. Under reasonable assumptions we show that our new protocol is more efficient than the recent protocol of Abdalla and Pointcheval (FC 2005), proven in the random oracle model. We also observe some limitations in the model due to Abdalla, Fouque and Pointcheval (PKC 2005) for proving security of such protocols.

**Keywords:** Key agreement, password authentication, three-party.

## 1 Introduction

A major goal of modern cryptography is to enable two or more users on an insecure (adversary controlled) network to communicate in a confidential manner and/or ensure that such communications are authentic. Symmetric key cryptographic tools are often used for such communications, due to their efficiency. However, due to the impracticality of every pair of users sharing a large secret key, public key and/or password based techniques are used to generate such a key when it is required. We focus on password-based key exchange, which is useful in situations where the secure storage of full length cryptographic keys is infeasible, such as in mobile environments. However, because of the short length of the password, special care must be taken when designing protocols to ensure that both the password and the key finally agreed remain secret.

One area of recent attention is password-based 3-party protocols with a formal security proof. These protocols enable two clients to exchange a secret key where each client shares a (different) password with a common server. Such protocols overcome the problem associated with 2-party password-based protocols (such as all of the password-based protocols being standardized in IEEE P1363.2 and ISO/IEC FDIS 11770-4) whereby a single user must hold as many passwords as there are parties with whom it wishes to communicate.

Although such protocols have received some attention in the literature, formal proofs have only recently been provided. Abdalla, Fouque and Pointcheval [AFP05] proved the security of a generic construction (called GPAKE) that uses any two-party authenticated key exchange protocol as well as a three party key distribution protocol, and combines them with a Diffie-Hellman key exchange authenticated using a message authentication code (MAC). They proved this construction secure in a new model (which we call the AFP model) based on the models of Bellare et al. [BR93, BR95, BPR00]. However, protocols constructed according to this method can be quite inefficient.

The AFP model contains two variants. The first, called the find-then-guess (FTG) model, is similar to existing models, since it allows Reveal queries (to disclose the session key of a requested instance to

---

\*Research funded by Australian Research Council through Discovery Project DP0345775.

†This is the full version of the extended abstract which appears in the 4<sup>th</sup> International Conference on Applied Cryptography and Network Security (ACNS'06), Lecture Notes in Computer Science volume 3989, pages 146–161, Springer 2006.

the adversary) and only one Test query (where the adversary must guess whether it was told the actual session key of a session it selected). The other variant is called the real-or-random (ROR) model, and disallows Reveal queries, but allows multiple Test queries, where the keys returned by the test queries are either all real or all random. It is shown that the ROR model is stronger than the FTG model when password-based protocols are being studied. However, when high-entropy keys are used rather than passwords, protocols secure in one variant are secure in the other also.

The AFP model also defines a new notion, *key privacy*, which means that the server cannot deduce the value of the secret key shared between the clients. Key privacy may be proven separately to the protocol’s semantic security. However, the AFP model does have the shortcoming of not allowing adaptive corrupt queries; corrupted parties are chosen statically at the beginning of a proof in the AFP model.

The GPAKE protocol was proven secure in the ROR variant of the AFP model, assuming that the two-party authenticated key exchange protocol used with it is also secure in the ROR model. Although most suitable password based protocols have been proven secure in the FTG model, it is claimed that most proofs, including the KOY one [KOY01], can be modified easily to meet the ROR model requirements.

Abdalla and Pointcheval [AP05] later proposed another 3-party password-based protocol, to which we refer as the AP protocol. It was proven secure using the FTG variant of the AFP model, using the random oracle (RO) model (note that earlier versions, including the conference version, have an error in the protocol description that leads to an attack [CBH05b]). The proof requires new and stronger variants of the Decisional Diffie-Hellman (DDH) assumption. The authors claim that their protocol is quite efficient, requiring 2 exponentiations and a few multiplications per party, or less than half the cost for the server compared with using GPAKE.

In this paper we propose another 3-party password-based protocol, proven secure using the Canetti-Krawczyk (CK) proof model [CK01]. This model allows the adversary to make adaptive corrupt queries. In contrast, the AFP model only allows static corrupt queries. We therefore select the CK model as it can model a wider variety of attack scenarios and allows the modular design of protocols by enabling key exchange and authentication mechanisms to be proven secure separately.

We regard it as a significant advantage that our proof is in the standard model, in contrast to the AP protocol which requires the RO model. We also examine the AP protocol efficiency claims more closely and claim that our new protocol can be more efficient with reasonable assumptions.

The rest of this paper proceeds as follows. Section 2 reviews the CK model, and is followed by a description of the protocol and its security proof in Section 3. Section 4 provides the proof of a slight modification to the password-based authenticator, and Section 5 discusses the efficiency of signature and encryption schemes with proofs in the standard model that can be used in conjunction with the CK authenticators. Finally, Section 6 discusses the efficiency, advantages and disadvantages of the proposed scheme in comparison to the AP and GPAKE protocols.

## 2 The Canetti–Krawczyk Model

In this section the CK approach is reviewed. Further details of the model can be found in the original papers [BCK98, CK01] and a paper extending the model to justify optimization techniques [HBGN05].

### 2.1 Key Establishment Protocols

In the CK model a protocol  $\pi$  is modeled as a collection of  $n$  programs running at different parties,  $P_1, \dots, P_n$ . Each program is an interactive probabilistic polynomial-time (PPT) machine. Each invocation of  $\pi$  within a party is defined as a *session*, and each party may have multiple sessions running concurrently. The communications network is controlled by an adversary  $\mathcal{A}$ , also a PPT machine, which schedules and mediates all sessions between the parties. When first invoked within a party, a key exchange protocol  $\pi$  calls an initialization function that returns any information needed for the bootstrapping of the cryptographic authentication functions (e.g. private keys and authentic distribution of other parties’ public keys). After this initialization stage, the party waits for activation.  $\mathcal{A}$  may activate a party  $P_i$  in two ways:

1. by means of an `establish-session( $P_i, P_j, s, role$ )` request, where  $P_j$  is another party with whom the key is to be established,  $s$  is a session-id string which uniquely identifies a session between the

participants, and  $role \in \{\text{initiator}, \text{responder}\}$ . Note that the session-id is chosen by the adversary, with the restriction that it must be unique among all sessions between the two parties involved. This allows the delivery of messages to the right protocol instantiation within a party.

2. by means of an *incoming message*  $m$  with a specified sender  $P_j$ .

A restriction exists on how the adversary activates parties, depending on which of the following adversarial models is being considered.

- The *authenticated-links model (AM)* defines an idealized adversary,  $\mathcal{A}$ , that is restricted to delivering messages faithfully between uncorrupted parties, if at all. That is, the adversary may only deliver messages where the specified (uncorrupted) sender actually sent that message to the specified recipient. The *AM-adversary* is not allowed to fabricate, modify, or replay messages of its choice except if the message is purported to come from a corrupted party. However, the adversary may decide not to deliver some messages from uncorrupted parties at all, or to deliver messages out of order.
- The *unauthenticated-links adversarial Model (UM)* is a more realistic model in which the adversary,  $\mathcal{U}$ , does not have the above restriction. Thus, a *UM-adversary* can fabricate messages and deliver any messages of its choice.
- the *hybrid model (HM)*, with adversary  $\mathcal{H}$ , combines the above models, and messages may be marked by the sender either authentic (so that AM restrictions apply to the message) or unauthentic (so that UM rules apply).

If no unauthentic messages are sent in the HM, it is the same as the AM. If no authentic messages are sent in the HM, it is the same as the UM.  $\mathcal{H}$  may fabricate unauthentic messages in the HM.

Upon activation, the parties perform some computations, update their internal state, and may output messages together with the identities of the intended receivers. Sessions may also add special messages to a local output to record the occurrence of important, security-related events. For example, when the run of a session finishes, a special key establishment event  $(P_i, P_j, s, \kappa)$  is recorded in the local output, to denote that a key  $\kappa$  has been established with party  $P_j$ .  $\kappa = \text{null}$  denotes that the session has been aborted. When  $\kappa \neq \text{null}$  the session is said to be completed. The adversary's view consists of all parties' public authentication information, output messages and local outputs, except for the established keys ( $\kappa$ -values) of completed sessions. Two sessions  $(P_i, P_j, s, \text{role})$  and  $(P'_i, P'_j, s', \text{role}')$  are said to be *matching sessions* if  $P_i = P'_j$ ,  $P_j = P'_i$ , and  $s = s'$ , i.e. if their session-ids are identical and they recognised each other as their respective communicating partner for the session.

In addition to the activation of parties,  $\mathcal{A}$  can perform the following queries:

1. *corrupt* $(P_i)$ . With this query  $\mathcal{A}$  learns the entire current state of  $P_i$  including long-term secrets, session internal states and session keys. From this point on,  $\mathcal{A}$  may issue any message in which  $P_i$  is specified as the sender and play the role of  $P_i$ . Whenever  $\mathcal{A}$  corrupts a party, the event is recorded in the local output of that party, and that party is never activated again;
2. *session-key* $(P_i, P_j, s)$ . This query returns the session key (if any) accepted by  $P_i$  during a given session  $s$  with  $P_j$ . This event is recorded in the local output of that party;
3. *session-state* $(P_i, P_j, s)$ . This query returns all the internal state information of party  $P_i$  associated to a particular session  $s$  with  $P_j$ . This event is recorded in the local output of that party and is sometimes known as session corruption (as opposed to party corruption above);
4. *session-expiration* $(P_i, P_j, s)$ . This query can only be performed on a completed session. It is used for defining *forward secrecy* and ensures that the corresponding session key is erased from the party's memory. The session is thereafter said to be expired. This event is recorded in the local output of that party;
5. *test-session* $(P_i, P_j, s)$ . To respond to this query, a random bit  $b$  is selected. If  $b = 1$  then the session key is output. Otherwise, a random key is output chosen from the probability distribution of keys generated by the protocol. This query can only be issued to a session that has not been *exposed*. A session is exposed if the adversary performs any of the following actions:

- a session-state or session-key query to this session or to the matching session, or
- a corrupt query to either partner before the session expires at that partner.

Security is defined based on a game played by the adversary. In this game  $\mathcal{A}$  interacts with the protocol. In a first phase of the game,  $\mathcal{A}$  is allowed to activate sessions and perform `corrupt`, `session-key`, `session-state` and `session-expiration` queries as described above. The adversary then performs a `test-session` query to a party and session of its choice. The adversary is not allowed to expose the test-session.  $\mathcal{A}$  may then continue with its regular actions with the exception that no more `test-session` queries can be issued. Eventually,  $\mathcal{A}$  outputs a bit  $b'$  as its guess on whether the returned value to the `test-session` query was the session key or a random value, then halts.  $\mathcal{A}$  wins the game if  $b = b'$ . The definition of security follows.

**Definition 1.** *A key establishment protocol  $\pi$  is called session key (SK-) secure with perfect forward secrecy (PFS) in the UM (resp. AM) if the following properties are satisfied for any UM (resp. AM) adversary  $\mathcal{A}$ .*

1. *If two uncorrupted parties complete matching sessions then they both output the same key;*
2. *The probability that  $\mathcal{A}$  guesses correctly the bit  $b$  is no more than  $\frac{1}{2}$  plus a negligible function in the security parameter.*

We define the advantage of  $\mathcal{A}$  to be twice the probability that  $\mathcal{A}$  wins, minus one. Hence the second requirement will be met if the advantage of  $\mathcal{A}$  is negligible. Canetti and Krawczyk also provide a definition of SK-security *without PFS*. The only difference with respect to the above definition is that now the adversary is not allowed to expire sessions.

## 2.2 Authenticators

Protocols that are SK-secure in the AM can be converted into SK-secure protocols in the UM by applying an *authenticator* to them. An authenticator is a compiler  $\mathcal{C}$  that takes as input a protocol  $\pi$  and outputs another protocol  $\pi' = \mathcal{C}(\pi)$ , with the property that if  $\pi$  is SK-secure in the AM, then  $\pi'$  is SK-secure in the UM. Authenticators are in fact defined to achieve the stronger notion of *protocol emulation*. The authenticated protocol  $\pi'$  emulates protocol  $\pi$  in the UM, which informally means that whatever a UM adversary can do against  $\pi'$  can be done by an AM adversary against  $\pi$ . It is defined by requiring that for any UM adversary there exist an AM adversary such that the outputs of all parties in the AM and UM are computationally indistinguishable. Definitions for *HM-authenticators* are similar, except that they map a protocol from the HM to the HM.

Authenticators can be constructed from *message transmission (MT) authenticators*. An MT-authenticator is a protocol that enables delivery of messages in the UM in an authenticated manner. However, each message is authenticated almost independently of all other messages. (Although the same long term keys may often be used by an authenticator to authenticate many messages, other data used by the authenticator such as a nonce or signature is generated separately for each message to be authenticated.) To translate an SK-secure protocol in the AM to an SK-secure protocol in the UM an MT-authenticator can be applied to each message and the resultant sub-protocols combined to form one overall SK-secure protocol in the UM. An MT-authenticator emulates the MT protocol, in which the sender  $A$  outputs ' $A$  sent  $m$  to  $B$ .' and sends  $(A, B, m)$  to party  $B$ , and upon receipt of the authentic message,  $B$  outputs ' $B$  received  $m$  from  $A$ .'

Constructing an authenticator by using an MT-authenticator for each message can lead to very inefficient protocols due to the large number of messages generated and the requirement that the session identifier be known before the protocol begins. Until recently, heuristic arguments were made as to why optimized versions of protocols where messages were shifted and nonces reused were secure. However, recent work [HBGN05] has provided a formal basis for such optimizations by showing and/or proving how to define a session identifier part way through the protocol, that more than one MT-authenticator may be used to construct an authenticator for an entire protocol, that certain *preamble* authenticator messages can be shifted to earlier points in the protocol if some conditions are met, that the message  $m$  being authenticated need not form part of every authenticator message, and that nonces used in some authenticators only need to be previously unused by that party in that authenticator and may be replaced with other values from the protocol. The techniques presented in that work will be used throughout this paper.

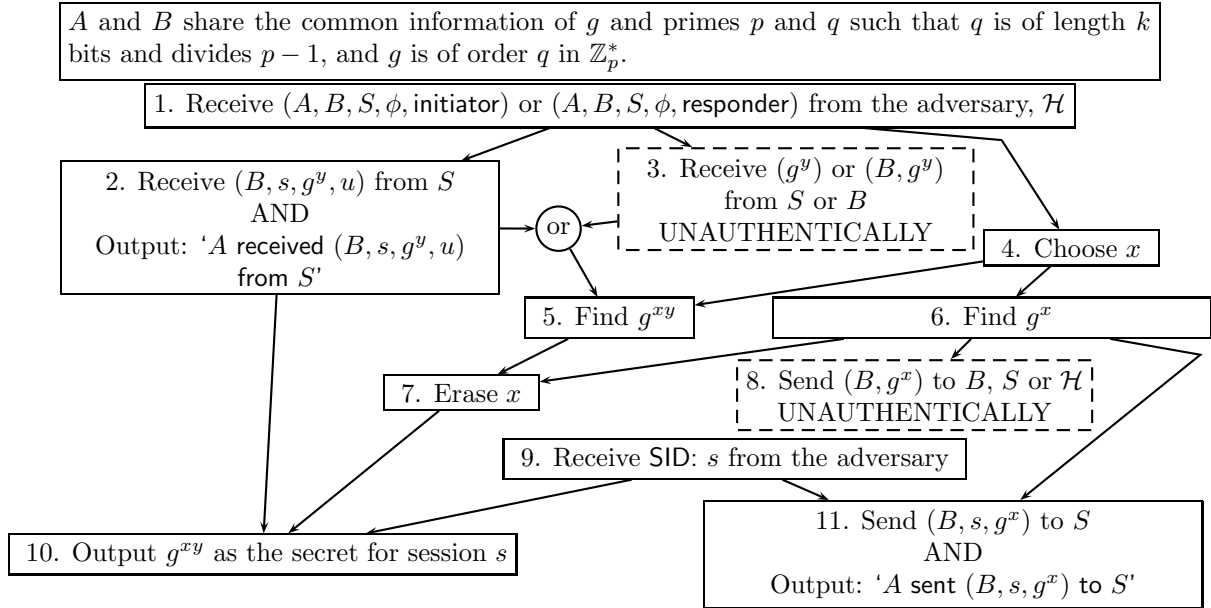


Figure 1: Possible step order for receiver and responder in the 3DH protocol in the HM

### 3 Conversion from Two-party to Three-party Protocol

In this section, we propose a new protocol, labelled 3DH, that uses a server’s assistance to perform the Diffie-Hellman key agreement between two parties. The protocol’s purpose is to enable the use of a password-based authenticator between each of the parties and the server,  $S$ . The server is a gateway responsible for connecting parties  $A$  and  $B$  faithfully and providing assurance of the identify of each party to the other.

Figure 1 shows an HM template describing the possible actions for party  $A$ , the initiator or responder of the protocol, and specifies which actions are prerequisites for others, and which may be performed in parallel. It also specifies which actions must be performed in a single activation. The use of such a template has been recommended [HBGN05] to clearly show the security requirements of a protocol and yet allow it to be easily adapted to suit different authenticators or objectives, without breaking the security proof.

In the diagram, all messages are assumed to be authentic, unless otherwise specified, and actions that must be performed in the same activation are shown in the one box and joined with “AND.” An arrow from one step to another indicates that the first step must be completed before the second is begun. Optional steps are shown using dashed boxes. The session identifier  $s$  received from the adversary must be the same as that received in the authentic message from  $B$ , otherwise the protocol halts without outputting a secret key. The template for party  $B$  is identical, except for the renaming of  $A$  to  $B$ ,  $B$  to  $A$ ,  $x$  to  $y$  and  $y$  to  $x$ . The value  $u$  received from  $S$  is not used in the template. Its purpose is described below.

In Figure 2, the possible interaction between the server and any party  $B$  is shown (i.e.  $B$  in Figure 2 may correspond to either  $A$  or  $B$  in Figure 1). In a single protocol run, such interaction may occur between the server and more than one party. The value  $u$  is included to ensure the requirement that all authentic messages are unique [BCK98, full version p.8, footnote 2] is met. Some authenticator proofs require this property.

An examination of Figures 1 and 2 shows that a number of steps can be performed if one party has possession of the other’s unauthentic Diffie-Hellman value. Therefore, it seems logical to specify an HM protocol such as the one shown by Protocol 1 that allows a party to receive an unauthentic Diffie-Hellman value and then carry out as many actions as possible. Messages not labelled “unauth’c” are authentic. In this version, the adversary chooses the Diffie-Hellman values,  $(g^x, g^y)$ , to be the session identifier, and the unauthentic message  $g^y$  from  $B$  to the HM adversary,  $\mathcal{H}$ , facilitates this. Later changes will allow the clients to choose the session identifier, so the messages in boxes will then be removed.

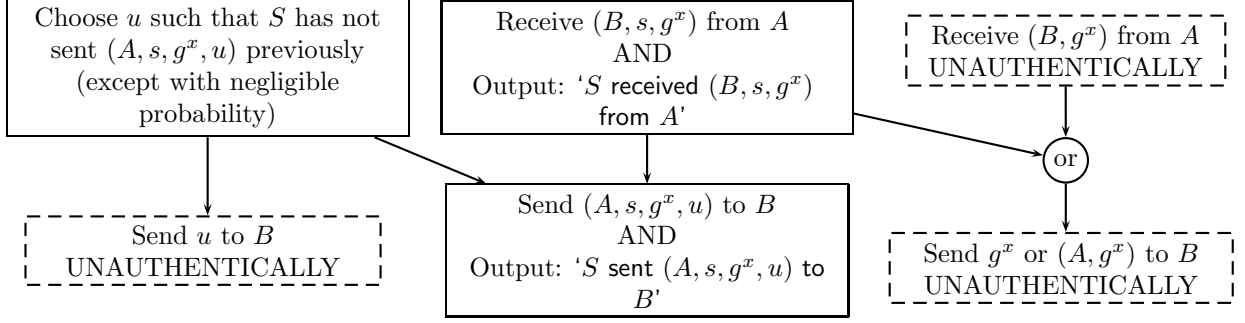
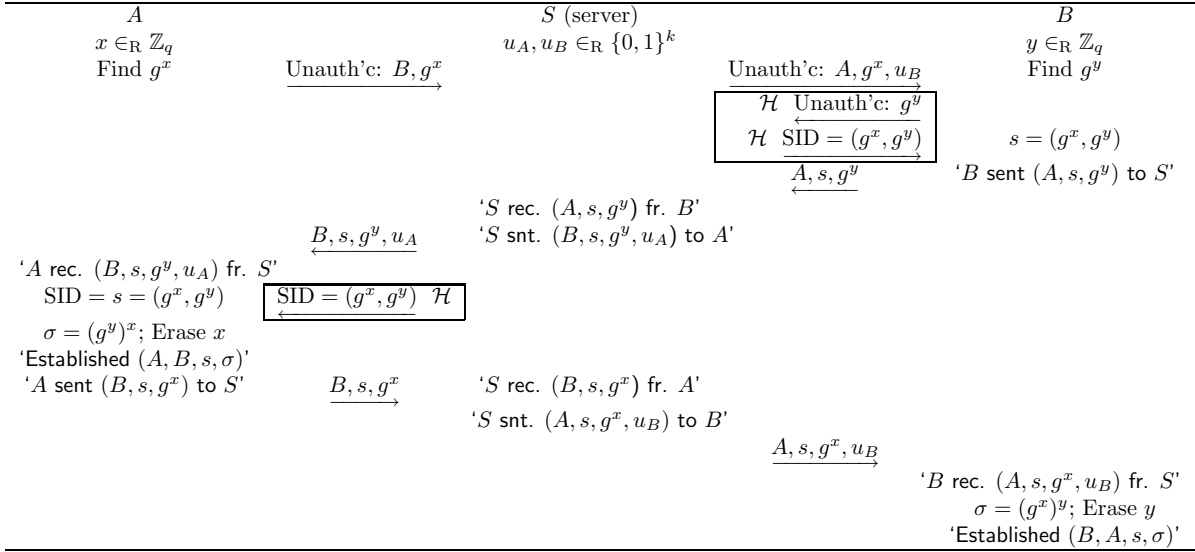


Figure 2: Possible step order for the server of the 3DH protocol in the HM, with optional steps in dashed boxes



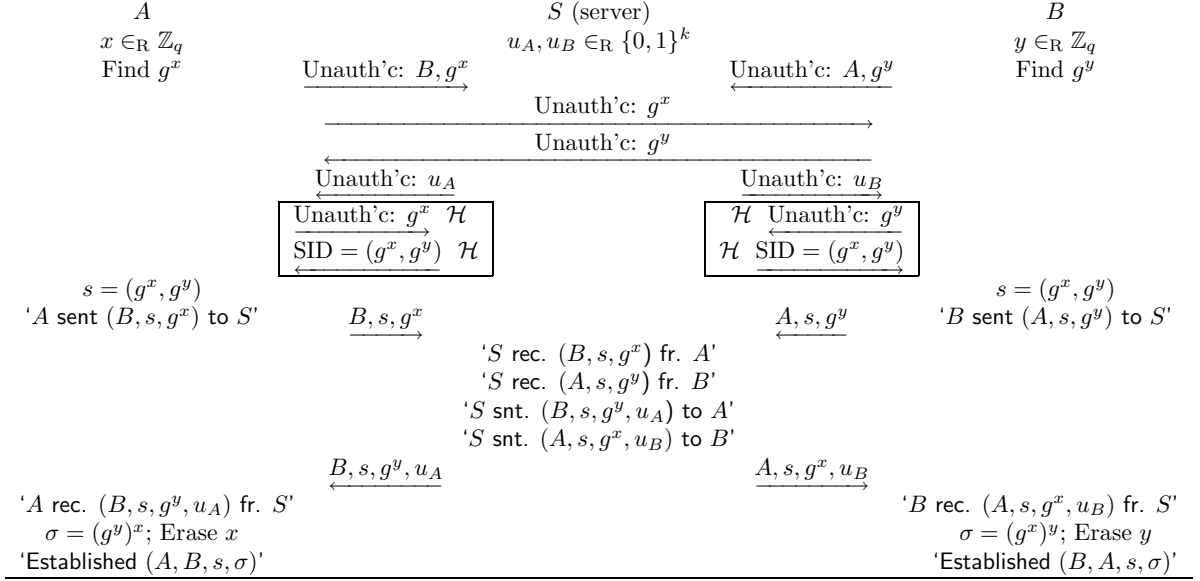
Protocol 1: A possible server aided 3DH HM protocol

Protocol 2 can also be derived from Figures 1 and 2, and is intended to allow a final protocol with fewer rounds than that associated with Protocol 1. The notation used in both protocols is the same, and the boxed messages will eventually be removed. Furthermore, the first six messages (which are unauthentic) constitute one round, and may be sent in any order. The final four messages require two rounds.

Our new protocol differs from the ordinary Diffie-Hellman key exchange in that protocol participants do not directly make contact with each other. Although password-based protocols where the participants communicate directly with one another do exist and have been proven secure in other proof models [KOY01, Mac02], they do not use a public key for the server and are not amenable to the CK-model, since their key exchange and authentication mechanisms cannot be separated [HK99]. Furthermore, such protocols require a party to maintain a list of passwords for each of the targets with whom it wishes to communicate. Thus the advantages of the modular approach used by the CK-model cannot be realized by such protocols, and they are inappropriate when storage constraints exist, such as those in wireless networks.

The proposed protocol uses a common trusted server for authentication, a common practice in networking. The server is unable to calculate the session key; it can only be generated by two authenticated clients. Moreover, client authentication uses passwords which can be memorized by users, eliminating the need for shared secrets to be stored in the users' devices, and greatly reducing the potential security risk.

We disallow server corruption because there is no way of guaranteeing the establishment of secure session keys using a corrupted server controlled by the adversary. A corrupted server would effectively



Protocol 2: A possible server aided 3DH HM protocol for fewer rounds

allow an adversary to inject authentic messages from the server into the network. In addition, session-state reveals on  $S$  have been disallowed to keep the partner and session identity definitions simple. However, if server session-state reveal queries were allowed, it would not affect the security of the protocol when used in conjunction with existing authenticators. This is because the server only forwards messages between the two clients. Since such messages are not secret, session-state reveals do not reveal any information that can be used effectively in an attack.

Proving the security of 3DH requires the use of the decisional Diffie-Hellman (DDH) assumption [Bon98]. The definition and proof are based in  $\mathbb{Z}_p^*$  but the protocol may also be run in an elliptic curve group where the elliptic curve version of the DDH assumption holds.

**Assumption 1 (Decisional Diffie-Hellman (DDH)).** *Let  $k$  be a security parameter. Let primes  $p$  and  $q$  be such that  $q$  is of length  $k$  bits and divides  $p-1$ , and let  $g$  be of order  $q$  in  $\mathbb{Z}_p^*$ . Then the probability distributions  $Q_0 = \{ \langle p, g, g^x, g^y, g^{xy} \rangle : x, y, \xleftarrow{R} \mathbb{Z}_q \}$  and  $Q_1 = \{ \langle p, g, g^x, g^y, g^z \rangle : x, y, z \xleftarrow{R} \mathbb{Z}_q \}$  of quintuples are computationally indistinguishable.*

**Theorem 1.** *Any Diffie-Hellman based HM protocol where the actions of each party satisfy the requirements specified in Figures 1 and 2 is SK-secure<sup>1</sup>, provided the DDH assumption holds.*

*Proof.* The proof is very similar to Canetti and Krawczyk's Theorem 8 [CK01] and Hitchcock et al.'s Theorem 4 [HBGN05]. Rather than pointing out those parts which are different to the existing proofs, which would be difficult to read for those unfamiliar with the existing proofs, the entire proof is given below.

To see that if two uncorrupted parties complete matching sessions then they both output the same key (requirement 1 of Definition 1), note that if both  $P_i$  and  $P_j$  are uncorrupted during the exchange of the key and both complete the protocol with matching sessions, then they both establish the same key, which is  $g^{xy}$ . This is because the session identifier  $s$  uniquely binds the values of  $g^x$  and  $g^y$  to these particular matching sessions and differentiates them from other exponentials that the parties may exchange in other (possibly simultaneous) sessions.

We now show that requirement 2 of Definition 1 is met. Assume to the contrary that there is an adversary  $\mathcal{H}$  in the HM against the protocol that has a non-negligible advantage in guessing correctly whether the response to a test-query is real or random. Out of this attacker  $\mathcal{H}$ , we construct an algorithm  $\mathcal{D}$  that distinguishes between the distributions  $Q_0$  and  $Q_1$  with non-negligible probability, thus reaching a contradiction with Assumption 1. The input to  $\mathcal{D}$  is denoted by  $(p, g, \alpha^*, \beta^*, \gamma^*)$  and is chosen from

<sup>1</sup>We use the 2-party SK-security definition, since  $S$  provides authentication only, cannot be corrupted or have its sessions revealed, and does not share the secret key.

$Q_0$  or  $Q_1$  each with probability  $1/2$ . Let  $l$  be an upper bound on the number of sessions invoked by  $\mathcal{H}$  for a particular party other than  $S$  in any interaction.  $\mathcal{D}$  uses  $\mathcal{H}$  as a subroutine and proceeds as follows:

1. Choose  $A, B \stackrel{R}{\leftarrow} \{P_1, P_2, \dots, P_n\}$  and  $r_A, r_B \stackrel{R}{\leftarrow} \{1, 2, \dots, l\}$ . We will refer to  $A$ 's  $r_A^{\text{th}}$  session and  $B$ 's  $r_B^{\text{th}}$  session the *target sessions*.
2. Invoke  $\mathcal{H}$  on a simulated interaction in the HM with parties  $S, P_1, \dots, P_n$  running the DH protocol. Hand  $\mathcal{H}$  the values  $p$  and  $g$  as the public parameters for the protocol execution.
3. Whenever  $\mathcal{H}$  activates a party to establish a new session (except for the target sessions) or to receive a message, follow the instructions of the server-aided DH protocol on behalf of that party. When a session is expired at a player erase the corresponding session key from that player's memory. When a party (other than  $A$  or  $B$  if that party's target session has not been expired) is corrupted or a session (other than a target session) is exposed, hand  $\mathcal{H}$  all the information corresponding to that party or session as in a real interaction.
4. When  $A$ 's  $r_A^{\text{th}}$  session is invoked within  $A$  to exchange a key with  $P_j$ , if  $B \neq P_j$ , then  $\mathcal{D}$  outputs  $b' \stackrel{R}{\leftarrow} \{0, 1\}$  and halts. Otherwise, whenever  $A$  would send the a message containing  $(g^x)$  to  $S$  (either authentically or unauthentically) let  $A$  send the same message but with  $(g^x)$  replaced with  $(\alpha^*)$  instead.
5. When  $B$ 's  $r_B^{\text{th}}$  session is invoked to exchange a key with  $P_i$ , if  $A \neq P_i$ , then  $\mathcal{D}$  outputs  $b' \stackrel{R}{\leftarrow} \{0, 1\}$  and halts. Otherwise, whenever  $B$  would send a message containing  $(g^y)$  to  $S$  (either authentically or unauthentically) let  $B$  send  $S$  the same message, but with  $(g^y)$  replaced with  $(\beta^*)$  instead.
6. If a target session is chosen by  $\mathcal{H}$  as the test-session, and both the target sessions have the same session identifier  $s$ , then provide  $\mathcal{H}$  with  $\gamma^*$  as the answer to this query. (Whether the sessions have the same session identifier can be deduced from the outputs of the session. A session matching the test session must exist, since completion of the test session means that it has received an authentic message containing the session identifier from the other party. Furthermore, that session identifier must be in the other party's output. If either of the target sessions has not produced any output containing its session identifier, then it is not a partner of the test session.)
7. If a target session is ever exposed, or a session other than a target session is chosen as the test-session, or if the test session does not match either of the target sessions, or if  $\mathcal{H}$  halts without choosing a test-session then  $\mathcal{D}$  outputs  $b' \stackrel{R}{\leftarrow} \{0, 1\}$  and halts.
8. If  $\mathcal{H}$  halts and outputs a bit  $b'$ , then  $\mathcal{D}$  halts and outputs  $b'$  too.

First note that the run of  $\mathcal{H}$  by  $\mathcal{D}$  up to the point where  $\mathcal{H}$  stops (or  $\mathcal{D}$  aborts  $\mathcal{H}$ 's run) is identical to a normal run of  $\mathcal{H}$  against the DH protocol.

Secondly, note that in step 3, it is always possible for  $\mathcal{D}$  to answer the corrupt and exposure queries of  $\mathcal{H}$ , unless step 7 specifies that  $\mathcal{D}$  should output a random bit and halt. This is so since if  $\mathcal{D}$  is not to output a random bit and halt:

- if any session is exposed, it must not be a target session. Therefore the discrete logarithms of  $\alpha^*$  and  $\beta^*$  will not be part of the exposed session's state in any way. Also  $\gamma^*$  will not be part of the session's state.
- if a target session is corrupted, the corrupted session must first have been expired. However, if the session was expired, it must also have been completed, and if it was completed, then the discrete logarithm of the exponential generated by the session ought to have been erased. Therefore, the discrete logarithms of  $\alpha^*$  and  $\beta^*$  will not be part of the corrupted session's state in any way.  $\gamma^*$  will not be part of the state either, since the expiry erased the session key.

Consider the case in which the test session chosen by  $\mathcal{H}$  coincides with a target session chosen at random by  $\mathcal{D}$ . In this case, the response to the test-query by  $\mathcal{H}$  is  $\gamma^*$ . Thus, if the input to  $\mathcal{D}$  came from  $Q_0$  then the response was the actual value of the key exchanged between  $P_i$  and  $P_j$  during the test-session (since, by construction, the session key exchanged in Steps 4 and 5 of the actions of  $\mathcal{D}$  is

$A$	$S$ (server)	$B$
$x \in_{\mathbb{R}} \mathbb{Z}_q$ Find $g^x$	$u_A, u_B \in_{\mathbb{R}} \{0, 1\}^k$	$y \in_{\mathbb{R}} \mathbb{Z}_q$
	$\xrightarrow{\text{Unauth}'c: B, g^x}$	
		$\xrightarrow{\text{Unauth}'c: A, g^x, u_B}$
	‘ $S$ rec. $(A, g^x, g^y)$ fr. $B$ ’	$\xleftarrow{A, g^x, g^y}$
	$\xleftarrow{B, g^x, g^y, u_A}$	‘ $B$ snt. $(A, g^x, g^y)$ to $S$ ’
‘ $A$ rec. $(B, g^x, g^y, u_A)$ fr. $S$ ’ Set $SID = s = (g^x, g^y)$ $\sigma = (g^y)^x$ ; Erase $x$ ‘Established $(A, B, s, \sigma)$ ’	‘ $S$ snt. $(B, g^x, g^y, u_A)$ to $A$ ’	
‘ $A$ snt. $(B, g^y, g^x)$ to $S$ ’	$\xrightarrow{B, g^y, g^x}$	
	‘ $S$ rec. $(B, g^y, g^x)$ fr. $A$ ’	
	‘ $S$ snt. $(A, g^y, g^x, u_B)$ to $B$ ’	
		$\xrightarrow{A, g^y, g^x, u_B}$
		‘ $B$ rec. $(A, g^y, g^x, u_B)$ fr. $S$ ’ $\sigma = (g^x)^y$ ; Erase $y$ ‘Established $(B, A, s, \sigma)$ ’

Protocol 3: Secure 3DH protocol suitable for optimization

$\gamma^* = g^{xy}$ ). On the other hand, if the input to  $\mathcal{D}$  came from  $Q_1$  then the response to the test query was a random exponentiation, i.e. a random value from the distribution of keys generated by the protocol. In addition, the input to  $\mathcal{D}$  was chosen with probability  $1/2$  from  $Q_0$  and with probability  $1/2$  from  $Q_1$ , so the distribution of responses provided by  $\mathcal{D}$  to the test query of  $\mathcal{H}$  is the same as specified by Definition 1. In this case, the probability that  $\mathcal{H}$  guesses correctly whether the test value was “real” or “random” is  $1/2 + \epsilon$  for non-negligible  $\epsilon$ . By the above argument this is equivalent to guessing whether the input to the distinguisher  $\mathcal{D}$  came from  $Q_0$  or  $Q_1$ , respectively. Thus, by outputting the same bit as  $\mathcal{H}$  we get that the distinguisher  $\mathcal{D}$  guesses correctly the input distribution  $Q_0$  or  $Q_1$  with the same probability  $1/2 + \epsilon$  as  $\mathcal{H}$  did.

Now consider the case in which a target session is not chosen as a test-session. In this case  $\mathcal{D}$  always ends outputting a random bit, and thus its probability to guess correctly the input distribution is  $1/2$ .

Since the first case (in which the test-session and a target session coincide) happens with probability  $\frac{1}{n^{2/2}}$  while the other case happens with probability  $1 - \frac{1}{n^{2/2}}$ , the overall probability of  $\mathcal{D}$  to guess correctly is  $1/2 + \frac{\epsilon}{n^{2/2}}$ , and thus  $\mathcal{D}$  succeeds in distinguishing  $Q_0$  from  $Q_1$  with non-negligible advantage.  $\square$

Although the original 2DH protocol of Canetti and Krawczyk [CK01] included the identity of the sender in the protocol messages, it has not been explicitly included in messages here. It is assumed that the identities of the sender and recipient of the message are provided automatically. For authenticated HM messages, such information must be provided in any case for the message to be authentic. For unauthenticated messages, if there is no other source for the information (such as a header field on the message) it may be necessary to add the information to the message in practice. However, it is omitted here for clarity. As can be seen above, this has not affected the proof.

Protocol 3 shows another version of the Diffie-Hellman protocol where the adversary no longer inputs the session identifier to the parties. In addition, messages containing the same term twice have had the second term removed, and the unauthentic message to the adversary has been removed. Protocol 4 shows the corresponding protocol for when a final protocol with fewer rounds is required. Again, the first six (unauthentic) messages constitute the first round and may be sent in any order. The final four messages constitute two rounds. The proof of the following theorem is very similar to that of Theorem 5 of Hitchcock et al. [HBGN05].

**Theorem 2.** *If Protocol 1 is secure then so is Protocol 3. If Protocol 2 is secure then so is Protocol 4.*

*Proof.* For convenience, let  $\pi$  denote Protocol 1 (respectively, Protocol 2) and  $\pi'$  denote Protocol 3 (respectively, Protocol 4) throughout the proof. Protocol  $\pi'$  satisfies the first requirement of Definition 1 since the same key is calculated in  $\pi'$  and  $\pi$ . Therefore, it remains to show that the second requirement is satisfied. Suppose to the contrary that there exists an adversary  $\mathcal{H}'$  against  $\pi'$  with a non-negligible advantage. Then we construct an adversary  $\mathcal{H}$  against  $\pi$  that has a non-negligible advantage.  $\mathcal{H}$  runs

A	S (server)	B
$x \in_{\mathbb{R}} \mathbb{Z}_q$ Find $g^x$	$u_A, u_B \in_{\mathbb{R}} \{0, 1\}^k$	$y \in_{\mathbb{R}} \mathbb{Z}_q$ Find $g^y$
	Unauth'c: $B, g^x$ $\rightarrow$	Unauth'c: $A, g^y$ $\leftarrow$
	Unauth'c: $g^x$ $\rightarrow$	
	Unauth'c: $g^y$ $\rightarrow$	
	Unauth'c: $u_A$ $\leftarrow$	Unauth'c: $u_B$ $\leftarrow$
Set SID = $s = (g^x, g^y)$ 'A snt. $(B, g^y, g^x)$ to $S'$	$B, g^y, g^x$ $\rightarrow$	$A, g^x, g^y$ $\leftarrow$
	'S rec. $(B, g^y, g^x)$ fr. $A'$	'S rec. $(A, g^x, g^y)$ fr. $B'$
	'S snt. $(B, g^x, g^y, u_A)$ to $A'$	'S snt. $(A, g^y, g^x, u_B)$ to $B'$
	$B, g^x, g^y, u_A$ $\leftarrow$	$A, g^y, g^x, u_B$ $\rightarrow$
'A rec. $(B, g^x, g^y, u_A)$ fr. $S'$ $\sigma = (g^y)^x$ ; Erase $x$ 'Established $(A, B, s, \sigma)$		'B rec. $(A, g^y, g^x, u_B)$ fr. $S'$ $\sigma = (g^x)^y$ ; Erase $y$ 'Established $(B, A, s, \sigma)$

Protocol 4: Secure 3DH protocol suitable for optimization with fewer rounds

$\mathcal{H}'$  and simulates an interaction of  $\mathcal{H}'$  with imaginary parties  $P'_1, P'_2, \dots, P'_n$  running  $\pi'$ . The simulation proceeds as follows:

- Whenever  $\mathcal{H}'$  activates  $P'_i$  to start the protocol with  $P'_j$  as an initiator (or responder respectively),  $\mathcal{H}$  activates the corresponding party,  $P_i$ , in its own model to start a protocol run with  $P_j$  as the initiator (or responder respectively).
- Whenever a party  $P_i$  outputs an unauthentic message,  $\mathcal{H}$  causes  $P'_i$  to output the same unauthentic message to  $\mathcal{H}'$ , except in the case where the unauthentic message output by  $P_i$  was addressed to  $\mathcal{H}$ .
- Whenever a party  $P_i$  outputs an authentic message  $(P_j, s, g^x)$  to  $S$  where  $s = (g^x, g^y)$ ,  $\mathcal{H}$  causes  $P'_i$  to pass  $(P_j, g^y, g^x)$  to  $\mathcal{H}'$  for delivery as an authentic message to  $S$ .
- Whenever a party  $P_j$  outputs an authentic message  $(P_i, s, g^y)$  to  $S$  where  $s = (g^x, g^y)$ ,  $\mathcal{H}$  causes  $P'_j$  to pass  $(P_i, g^x, g^y)$  to  $\mathcal{H}'$  for delivery as an authentic message to  $S$ .
- Whenever the server  $S$  outputs an authentic message  $(P_j, s, g^y, u)$  to  $P_i$  where  $s = (g^x, g^y)$ ,  $\mathcal{H}$  causes  $S'$  to pass  $(P_j, g^x, g^y, u)$  to  $\mathcal{H}'$  for delivery as an authentic message to  $P'_i$ .
- Whenever the server  $S$  outputs an authentic message  $(P_i, s, g^x, u)$  to  $P_j$  where  $s = (g^x, g^y)$ ,  $\mathcal{H}$  causes  $S'$  to pass  $(P_i, g^y, g^x, u)$  to  $\mathcal{H}'$  for delivery as an authentic message to  $P'_j$ .
- Whenever an unauthentic message is delivered to a party  $P'_j$  or to  $S'$  by  $\mathcal{H}'$ ,  $\mathcal{H}$  delivers the same unauthentic message to  $P_j$  or to  $S$  respectively.
- Whenever an authentic message  $(P'_i, g^w, g^z)$  is delivered to the server  $S'$  from  $P'_j$  by  $\mathcal{H}'$ , if  $P_j$  is a responder,  $\mathcal{H}$  delivers  $(P_i, s, g^z)$  where  $s = (g^w, g^z)$  as an authentic message from  $P_j$  to  $S$ . Otherwise, since  $P_j$  is an initiator,  $\mathcal{H}$  delivers  $(P_i, s, g^z)$  where  $s = (g^z, g^w)$  as an authentic message from  $P_j$  to  $S$ .
- Whenever an authentic message  $(P'_j, g^w, g^z, u)$  is delivered from the server  $S'$  to  $P'_i$  by  $\mathcal{H}'$ , if  $P_i$  is an initiator,  $\mathcal{H}$  delivers  $(P_j, s, g^z, u)$  where  $s = (g^w, g^z)$  as an authentic message from  $S$  to  $P_i$ . Otherwise, since  $P_i$  is a responder,  $\mathcal{H}$  delivers  $(P_j, s, g^z, u)$  where  $s = (g^z, g^w)$  as an authentic message from  $S$  to  $P_i$ .
- Whenever a party  $P_i$  is waiting for a session identifier to be input,  $\mathcal{H}$  calculates the session identifier in the same way as in  $\pi$  and inputs it to  $P_i$ .
- Whenever a party  $P_i$  outputs that it has established a session key,  $\mathcal{H}$  causes  $P'_i$  to output the same thing.

- Whenever  $\mathcal{H}'$  chooses a session as the test session,  $\mathcal{H}$  chooses the corresponding session (i.e. the one with the same session identifier and run by the corresponding party) in its own model and passes the value it receives in response to its test session query to  $\mathcal{H}'$ .
- Whenever  $\mathcal{H}'$  corrupts a party or a session, or performs a session key query,  $\mathcal{H}$  corrupts the corresponding party or session or performs a session key query on the corresponding session in its own model and passes whatever it receives to  $\mathcal{H}'$ .
- Whenever  $\mathcal{H}'$  expires a session,  $\mathcal{H}$  expires the corresponding session in its own model.
- $\mathcal{H}$  outputs whatever  $\mathcal{H}'$  outputs.

Observe that any session  $s'$  with which  $\mathcal{H}'$  interacts will have the same session identifier and secret key (if they have been set) as the corresponding session  $s$  with which  $\mathcal{H}$  interacts. Furthermore, the matching session to  $s$  will correspond to the matching session to  $s'$  (if it exists). In addition, no two sessions belonging to one party will have the same session identifier (this is a requirement of the HM). Hence,  $\mathcal{H}$  will have the same advantage against Protocol 1 (respectively, Protocol 2) as  $\mathcal{H}'$  has against Protocol 3 (respectively, Protocol 4).  $\square$

We now focus on the authenticators to be used in conjunction with Protocol 3. The password-based authenticator,  $\lambda_{\text{P-ENC}}$ , shown in Protocol 3, has been chosen, so that clients do not need keep secret a large key. It has a proof [HTB<sup>+</sup>03] of password-based session key (PBSK-) security, when the encryption scheme is indistinguishable under chosen ciphertext attack (IND-CCA secure),  $H(m) \stackrel{\text{def}}{=} m$ ,  $\mathcal{E}_e$  denotes encryption with key  $e$  and  $\mathcal{D}_d$  denotes decryption with key  $d$ . This means that if the server refuses to complete sessions with a client after  $\gamma$  unsuccessful login attempts for that client, then the adversary has an advantage negligibly greater than the advantage due to simply guessing a password and attempting to impersonate the user online. The nonce  $N_B$  may be any value of  $B$ 's choice, including one chosen by the adversary, if it has not been used as a nonce in this authenticator before, except with negligible probability. Since  $N_B$  is a preamble message it may be sent before the first authenticator message.

In order to increase the efficiency of the authenticator (and hence the resulting protocols) we now alter the authenticator slightly, replacing the message  $m$  by  $H(m)$ . The authenticator is still PBSK-secure if  $H$  is chosen to be a collision resistant one-way hash function. This new authenticator is labelled  $\lambda_{\text{P-ENC-H}}$ . The proof is deferred to Section 4.

Another authenticator is required for messages from the server to the clients. The only suitable existing authenticators are the signature based and the encryption based authenticators [BCK98] (denoted  $\lambda_{\text{SIG}}$  and  $\lambda_{\text{ENC}}$ ), as other available authenticators would require the clients to hold secret keys. If signature and encryption schemes are chosen that have proofs of security in the standard model, the encryption and signature schemes have about the same efficiency. Therefore, we use  $\lambda_{\text{ENC}}$ , shown in Protocol 6, to minimize the number of separate cryptographic primitives that must be implemented. We observe that  $c$  is a preamble message, but  $r_B$  must be erased in the same activation as  $c$  is decrypted for the authenticator to be secure [CBH05a]. Further details of the efficiency of these authenticators and the associated signature and encryption schemes having proofs in the standard model are in Section 5.

Protocol 7 shows the result of applying  $\lambda_{\text{P-ENC-H}}$  and  $\lambda_{\text{ENC}}$  to Protocol 3. Similarly, Protocol 8 shows the result of applying  $\lambda_{\text{P-ENC-H}}$  and  $\lambda_{\text{ENC}}$  to Protocol 4 for a version with fewer rounds.  $S$  has two encryption keys to keep the state of each authenticator independent, as required by the proof that two or more authenticators may be applied to the one protocol [HBGN05].

We can begin to improve Protocols 7 and 8 by removing the authentic message (i.e. “ $m$ ” in the authenticator description) from the first and second messages of each authenticator. Since the authentic message is still being delivered in the last message of each authenticator, this is not a problem [HBGN05]. The parts to be deleted have been boxed in Protocols 7 and 8.

The optimization process can now be completed by replacing  $a$  and  $b$  with  $u_A$  and  $u_B$  respectively to reduce the number of values generated and transmitted, and by moving messages and piggybacking them together. Values  $a$  and  $b$  may be replaced since they are only required to be not previously used as a nonce with  $\lambda_{\text{P-ENC-H}}$ . Since  $u_A$  and  $u_B$  have negligible probability of being generated previously, they may be used in place of  $a$  and  $b$  [HBGN05]. The four messages shifted to earlier points in the protocol (those containing only  $b$ ,  $c_{A2}$ ,  $a$  and  $c_{B2}$ ) are all preamble messages. The values  $u_A$  and  $u_B$  are already known to the adversary at the time they are used in place of  $a$  and  $b$ , so this requirement for shifting and replacing the nonces is also met. The final results are shown in Protocols 9 and 10.

$A$ (Client)	$\lambda_{\text{P-ENC}}$	$B$ (Server)
Known: Password, $\pi$ Public key of $B$ , $e_B$		Known: Password of $A$ , $\pi$ Public/private keys, $(e_B, d_B)$
$c = \mathcal{E}_{e_B}(H(m), N_B, A, \pi)$	$\xrightarrow{m}$ $\xleftarrow{m, N_B}$ $\xrightarrow{m, c}$	$N_B \in_R \{0, 1\}^k$ $v = \mathcal{D}_{d_B}(c)$ Check $v \stackrel{?}{=} (H(m), N_B, A, \pi)$
$A$ (Server)	$\lambda_{\text{ENC}}$	$B$ (Client)
Known: Public/private keys, $(e_A, d_A)$		Known: Public key of $A$ , $e_A$
'A sent $m$ to $B$ ' $r_B = \mathcal{D}_{d_A}(c)$ $z = \mathcal{M}_{r_B}(m, B)$ ; Erase $r_B$	$\xrightarrow{m}$ $\xleftarrow{m, c}$ $\xrightarrow{m, z}$	$r_B \in_R \{0, 1\}^k$ $c = \mathcal{E}_{e_A}(r_B)$ Check $z \stackrel{?}{=} \mathcal{M}_{r_B}(m, B)$ 'B received $m$ from A'

Protocols 5 and 6: Password and encryption based authenticators

## 4 Proof of Security of $\lambda_{\text{P-ENC-H}}$

We begin by recalling the theorem stating the security of  $\lambda_{\text{P-ENC}}$  from the original proposal [HTB<sup>+</sup>03] in which  $H$  is the identity function. We follow this with a theorem and proof showing that the authenticator  $\lambda_{\text{P-ENC-H}}$  is secure, in which function  $H$  in  $\lambda_{\text{P-ENC}}$  is defined to be any collision resistant one-way hash function.

**Theorem 3 ([HTB<sup>+</sup>03]).** *Assume that the encryption scheme in use (where  $\mathcal{E}_e$  denotes encryption with key  $e$  and  $\mathcal{D}_d$  denotes decryption with key  $d$ ) is secure in the sense of being indistinguishable under chosen ciphertext attack (IND-CCA secure). Also assume that  $\gamma$  is the maximum number of unsuccessful attempts per client to complete the protocol with a server,  $H(m) \stackrel{\text{def}}{=} m$ ,  $k$  is a security parameter,  $c(k)$  is the number of clients in the model,  $s(k)$  is the number of servers in the model, the set of clients and the set of servers are disjoint sets, and the passwords are randomly chosen from a dictionary  $\mathcal{D}$  of size  $|\mathcal{D}|$ . Then the output of protocol  $\lambda_{\text{P-ENC}}$  is computationally indistinguishable from that of an MT-protocol in unauthenticated networks with probability  $(1 - \epsilon(k))$  where:*

$$\epsilon(k) \leq \left( \epsilon_0(k) + \left( 1 - \left( 1 - \frac{\gamma + 1}{|\mathcal{D}|} \right)^{s(k)c(k)} \right) \right) \quad (1)$$

and  $\epsilon_0(k)$  is negligible.

**Theorem 4.** *Theorem 3 still holds if  $H(m)$  is instead defined to be a collision resistant one-way hash function.*

*Proof.* Let the probability that the output of  $\lambda_{\text{P-ENC-H}}$  is computationally indistinguishable from that of an MT-protocol in unauthenticated networks be:

$$\epsilon^{\text{H}}(k) \leq \left( \epsilon_0^{\text{H}}(k) + \left( 1 - \left( 1 - \frac{\gamma + 1}{|\mathcal{D}|} \right)^{s(k)c(k)} \right) \right). \quad (2)$$

We wish to prove that  $\epsilon_0^{\text{H}}(k)$  is negligible for any adversary  $\mathcal{U}'$ . Suppose to the contrary that there exists an adversary  $\mathcal{U}'$  which interacts with  $\lambda_{\text{P-ENC-H}}$  such that  $\epsilon_0^{\text{H}}(k)$  is not negligible. Then we show that there exists an adversary  $\mathcal{U}$  against  $\lambda_{\text{P-ENC}}$  such that  $\epsilon_0(k)$  is also not negligible, contradicting Theorem 3.

We construct  $\mathcal{U}$  by having it run an internal copy of  $\mathcal{U}'$ .  $\mathcal{U}$  simulates the imaginary parties  $P'_1, P'_2, \dots, P'_n$  with whom  $\mathcal{U}'$  interacts as follows:



A	S (server)	B
Known: password, $\pi_A$ Public keys of S : $e_{S1}, e_{S2}$	Known: passwords $\pi_A, \pi_B$ Secret keys: $d_{S1}, d_{S2}$	Known: password, $\pi_B$ Public keys of S : $e_{S1}, e_{S2}$
$x \in_{\mathbb{R}} \mathbb{Z}_q; r_A \in_{\mathbb{R}} \{0, 1\}^k$ Find $g^x$	$a, b, u_A, u_B \in_{\mathbb{R}} \{0, 1\}^k$	$y \in_{\mathbb{R}} \mathbb{Z}_q; r_B \in_{\mathbb{R}} \{0, 1\}^k$ Find $g^y$
	$\xrightarrow{g^x}$	$\xleftarrow{A, g^y}$
	$\xleftarrow{g^y}$	
Set SID = $s = (g^x, g^y)$ 'A snt. (B, s) to S'		Set SID = $s = (g^x, g^y)$ 'B snt. (A, s) to S'
	$\xleftarrow{u_A}$	$\xrightarrow{u_B}$
	$\xrightarrow{\boxed{B, s}}$	$\xleftarrow{\boxed{A, s}}$
	$\xrightarrow{\boxed{B, s}, a}$	$\xrightarrow{\boxed{A, s}, b}$
$c_{A1} = \mathcal{E}_{e_{S1}}(H(B, s), a, A, \pi_A)$	$v_A = \mathcal{D}_{d_{S1}}(c_{A1})$	$c_{B1} = \mathcal{E}_{e_{S1}}(H(A, s), b, B, \pi_B)$
	$\xrightarrow{B, s, c_{A1}}$	$\xleftarrow{A, s, c_{B1}}$
	$v_A \stackrel{?}{=} (H(B, s), a, A, \pi_A)$ Erase $v_A$ 'S rec. (B, s) fr. A'	
	$v_B = \mathcal{D}_{d_{S1}}(c_{B1})$	
	$v_B \stackrel{?}{=} (H(A, s), b, B, \pi_B)$ Erase $v_B$ 'S rec. (A, s) fr. B'	
	'S snt. (B, s, $u_A$ ) to A'	
	'S snt. (B, s, $u_B$ ) to B'	
	$\xrightarrow{\boxed{B, s, u_A}}$	$\xrightarrow{\boxed{A, s, u_B}}$
$c_{A2} = \mathcal{E}_{e_{S2}}(r_A)$	$r_A = \mathcal{D}_{d_{S2}}(c_{A2})$	$c_{B2} = \mathcal{E}_{e_{S2}}(r_B)$
	$\xrightarrow{\boxed{B, s, u_A}, c_{A2}}$	$\xleftarrow{\boxed{A, s, u_B}, c_{B2}}$
	$z_A = \mathcal{M}_{r_A}(B, s, u_A, A)$ Erase $r_A$	
	$r_B = \mathcal{D}_{d_{S2}}(c_{B2})$	
	$z_B = \mathcal{M}_{r_B}(A, s, u_B, B)$ Erase $r_B$	
$z_A \stackrel{?}{=} \mathcal{M}_{r_A}(B, s, u_A, A)$	$\xleftarrow{B, s, u_A, z_A}$	$\xrightarrow{A, s, u_B, z_B}$
'A rec. (B, s, $u_A$ ) fr. S'		$z_B \stackrel{?}{=} \mathcal{M}_{r_B}(A, s, u_B, B)$
$\sigma = (g^y)^x$ ; Erase $x$		'B rec. (B, s, $u_B$ ) fr. S'
'Established (A, B, s, $\sigma$ )'		$\sigma = (g^x)^y$ ; Erase $y$
		'Established (A, B, s, $\sigma$ )'

Protocol 8: Unoptimized authenticated server aided 3DH for fewer rounds

- Whenever a party  $P_i$  outputs a message and encryption,  $(H(m), c)$ , for delivery to  $P_j$ ,  $\mathcal{U}$  causes  $P'_i$  to output a message and encryption  $(m, c)$  to  $\mathcal{U}'$  for delivery to  $P'_j$ . (Again, note that since  $\mathcal{U}$  provided  $H(m)$  to  $P_i$  after hashing it,  $\mathcal{U}$  is able to determine  $m$  from  $H(m)$  provided it keeps a list of all messages processed so far and their hashes.)
- Whenever  $\mathcal{U}'$  delivers the third authenticator message,  $(m, c)$ , to some party  $P'_j$  with the sender specified as  $P'_i$ ,  $\mathcal{U}$  delivers  $(H(m), c)$  to  $P_j$  with the sender specified as  $P_i$ . If  $P'_j$  generates some output (such as ' $P_j$  received  $H(m)$  from  $P'_i$ ') then  $\mathcal{U}$  causes  $P'_j$  to generate the same output, but with  $H(m)$  replaced with  $m$ .
- When a session is expired by  $\mathcal{U}'$  at a player  $P'_i$ , expire the corresponding session at  $P_i$ . When a party  $P'_i$  is corrupted or a session belonging to a party  $P'_i$  is exposed, hand  $\mathcal{U}'$  all the information corresponding to  $P_i$  or to the session at  $P_i$ , having first replaced  $H(m)$  with  $m$  wherever it occurs in the information.

Now, by construction, the probability that the output of  $\lambda_{\text{P-ENC}}$  is computationally indistinguishable from that of an MT-protocol in unauthenticated networks is equal to the probability that the output of  $\lambda_{\text{P-ENC-H}}$  is computationally indistinguishable from that of an MT-protocol in unauthenticated networks, except in the case where two messages are used which hash to the same value, which occurs with negligible probability. Hence, if  $\epsilon_0^{\text{H}}(k)$  is non negligible, so is  $\epsilon_0(k)$ , but this contradicts Theorem 3. Therefore we conclude that  $\epsilon_0^{\text{H}}(k)$  is negligible.  $\square$

$A$	$S$ (server)	$B$
Known: password, $\pi_A$ Public keys of $S$ : $e_{S1}, e_{S2}$	Known: passwords $pi_A, \pi_B$ Secret keys: $d_{S1}, d_{S2}$	Known: password, $\pi_B$ Public keys of $S$ : $e_{S1}, e_{S2}$
$x \in_{\mathbb{R}} \mathbb{Z}_q; r_A \in_{\mathbb{R}} \{0, 1\}^k$ Find $g^x$ $c_{A2} = \mathcal{E}_{e_{S2}}(r_A)$	$u_A, u_B \in_{\mathbb{R}} \{0, 1\}^k$	$y \in_{\mathbb{R}} \mathbb{Z}_q; r_B \in_{\mathbb{R}} \{0, 1\}^k$ Find $g^y$ Set $SID = s = (g^x, g^y)$ ' $B$ snt. $(A, s)$ to $S$ ' $c_{B1} = \mathcal{E}_{e_{S1}}(H(A, s), u_B, B, \pi_B)$ $c_{B2} = \mathcal{E}_{e_{S2}}(r_B)$
	$v_B = \mathcal{D}_{d_{S1}}(c_{B1})$ $v_B \stackrel{?}{=} (H(A, s), u_B, B, \pi_B)$ Erase $v_B$ ' $S$ rec. $(A, s)$ fr. $B$ ' ' $S$ snt. $(B, s, u_A)$ to $A$ ' $r_A = \mathcal{D}_{d_{S2}}(c_{A2})$ $z_A = \mathcal{M}_{r_A}(B, s, u_A, A)$ Erase $r_A$	
	$\xrightarrow{B, s, u_A, z_A}$	
$z_A \stackrel{?}{=} \mathcal{M}_{r_A}(B, s, u_A, A)$ ' $A$ rec. $(B, s, u_A)$ fr. $S$ ' Set $SID = s = (g^x, g^y)$ $\sigma = (g^y)^x$ ; Erase $x$ 'Established $(A, B, s, \sigma)$ ' Let $t = (g^y, g^x)$ ' $A$ snt. $(B, t)$ to $S$ ' $c_{A1} = \mathcal{E}_{e_{S1}}(H(B, t), u_A, A, \pi_A)$	$v_A = \mathcal{D}_{d_{S1}}(c_{A1})$ $v_A \stackrel{?}{=} (H(B, t), u_A, A, \pi_A)$ Erase $v_A$ ' $S$ rec. $(B, t)$ fr. $A$ ' ' $S$ snt. $(B, t, u_B)$ to $B$ ' $r_B = \mathcal{D}_{d_{S2}}(c_{B2})$ $z_B = \mathcal{M}_{r_B}(A, t, u_B, B)$ Erase $r_B$	$\xrightarrow{A, t, u_B, z_B}$ Let $t = (g^y, g^x)$ $z_B \stackrel{?}{=} \mathcal{M}_{r_B}(A, t, u_B, B)$ ' $B$ rec. $(B, t, u_B)$ fr. $S$ ' $\sigma = (g^x)^y$ ; Erase $y$ 'Established $(A, B, s, \sigma)$ '
	$\xrightarrow{B, t, c_{A1}}$	

Protocol 9: Optimized authenticated server aided Diffie–Hellman protocol

## 5 Efficiency of Secure Signature and Encryption Schemes

This section examines the efficiency of the signature based authenticator [BCK98],  $\lambda_{\text{SIG}}$ , shown in Protocol 11, and the encryption based authenticator [BCK98],  $\lambda_{\text{ENC}}$ , of Protocol 6. We observe that the nonce  $N_B$  may actually be any value of  $B$ 's choice in  $\lambda_{\text{SIG}}$ , including a value chosen by the adversary, provided that value has not previously been used as a nonce in  $\lambda_{\text{SIG}}$ , except with negligible probability. We also observe that  $N_B$  in the case of  $\lambda_{\text{SIG}}$ , and  $c$  in the case of  $\lambda_{\text{ENC}}$  are preamble messages and may therefore be sent before the first authenticator message. However,  $r_B$  must be erased in the same activation as  $c$  is decrypted for the authenticator to be secure [CBH05a].

Bellare et al. [BCK98] proved that  $\lambda_{\text{SIG}}$  is secure when used in conjunction with a signature scheme secure against adaptive chosen message attack. Such schemes that do not rely on the random oracle model for their proofs include the one by Cramer and Shoup [CS00], the one by Gennaro et al. [GHR99] (but note that Coron and Naccache [CN00] have attacked one variant, requiring an increase in the size of a security parameter), and the twin signature scheme by Naccache et al. [NPS01], which is based on Gennaro et al.'s scheme. Here we assume that the scheme of Cramer and Shoup will be used with  $\lambda_{\text{SIG}}$  because it has a good efficiency analysis and is at least comparable in efficiency to other schemes.

Bellare et al. [BCK98] also proved that  $\lambda_{\text{ENC}}$  is secure when used in conjunction with an encryption scheme that is semantically secure against chosen ciphertext attacks. Such a scheme that does not rely on the random oracle model for its proof is that of Cramer and Shoup [CS98]. In its basic form [CS98, Sect.

A	S (server)	B
Known: password, $\pi_A$ Public keys of $S$ : $e_{S1}, e_{S2}$	Known: passwords $\pi_A, \pi_B$ Secret keys: $d_{S1}, d_{S2}$	Known: password, $\pi_B$ Public keys of $S$ : $e_{S1}, e_{S2}$
$x \in_{\mathbb{R}} \mathbb{Z}_q; r_A \in_{\mathbb{R}} \{0, 1\}^k$ $c_{A2} = \mathcal{E}_{e_{S2}}(r_A)$ Find $g^x$	$u_A, u_B \in_{\mathbb{R}} \{0, 1\}^k$	$y \in_{\mathbb{R}} \mathbb{Z}_q; r_B \in_{\mathbb{R}} \{0, 1\}^k$ $c_{B2} = \mathcal{E}_{e_{S2}}(r_B)$ Find $g^y$
$\xrightarrow{B, g^x, c_{A2}}$	$\xrightarrow{g^x}$ $\xleftarrow{g^y}$	$\xleftarrow{A, g^y, c_{B2}}$
Set $SID = s = (g^x, g^y)$ 'A snt. $(B, s)$ to $S$ ' $c_{A1} = \mathcal{E}_{e_{S1}}(H(B, s), u_A, A, \pi_A)$ $\xrightarrow{B, s, c_{A1}}$	$v_A = \mathcal{D}_{d_{S1}}(c_{A1})$ $v_A \stackrel{?}{=} (H(B, s), u_A, A, \pi_A)$ Erase $v_A$ 'S rec. $(B, s)$ fr. A' $v_B = \mathcal{D}_{d_{S1}}(c_{B1})$ $v_B \stackrel{?}{=} (H(A, s), u_B, B, \pi_B)$ Erase $v_B$ 'S rec. $(A, s)$ fr. B' 'S snt. $(B, s, u_A)$ to A' 'S snt. $(B, s, u_B)$ to B' $r_A = \mathcal{D}_{d_{S2}}(c_{A2})$ $z_A = \mathcal{M}_{r_A}(B, s, u_A, A)$ Erase $r_A$ $r_B = \mathcal{D}_{d_{S2}}(c_{B2})$ $z_B = \mathcal{M}_{r_B}(A, s, u_B, B)$ Erase $r_B$	$\xleftarrow{A, s, c_{B1}}$ Set $SID = s = (g^x, g^y)$ 'B snt. $(A, s)$ to $S$ ' $c_{B1} = \mathcal{E}_{e_{S1}}(H(A, s), u_B, B, \pi_B)$ $\xrightarrow{A, s, c_{B1}}$ $z_B \stackrel{?}{=} \mathcal{M}_{r_B}(A, s, u_B, B)$
$z_A \stackrel{?}{=} \mathcal{M}_{r_A}(B, s, u_A, A)$ 'A rec. $(B, s, u_A)$ fr. S' $\sigma = (g^y)^x$ ; Erase $x$ 'Established $(A, B, s, \sigma)$ '	$\xleftarrow{B, s, u_A, z_A}$	$\xrightarrow{A, s, u_B, z_B}$ $z_B \stackrel{?}{=} \mathcal{M}_{r_B}(A, s, u_B, B)$ 'B rec. $(B, s, u_B)$ fr. S' $\sigma = (g^x)^y$ ; Erase $y$ 'Established $(A, B, s, \sigma)$ '

Protocol 10: Optimized authenticated server aided Diffie–Hellman protocol with 3 rounds

5.1], this scheme is much less efficient than the signature scheme, it only encrypts 1023 bit messages, and requires a modular square root to decode group elements into messages. As the proof of security requires the DDH assumption (amongst others), a suitable group must be used, such as a subgroup of order  $q$  of  $\mathbb{Z}_p^*$ , where  $p$  is a 1024 bit prime. Since messages must be elements of the subgroup in the basic scheme,  $q$  is chosen such that  $2q = p - 1$ . Messages between 0 and  $q - 1$  can be encoded into the subgroup by squaring modulo  $p$ .

The basic form of this encryption scheme is expected to suffice for use with  $\lambda_{\text{P-ENC-H}}$ , although it is fairly inefficient. Since  $\lambda_{\text{ENC}}$  only requires the encryption of a nonce, it may be possible to choose  $q$  to be smaller, provided a nonce may be a member of the subgroup. This would depend on whether the group element could be mapped to a key for the MAC function in a sufficiently secure way.

An efficiency improvement to the above basic scheme may be made by using a hybrid scheme [Sho00, CS03] (using symmetric techniques to encrypt the message and using public key techniques to encrypt the symmetric key), and the size of  $q$  need only be 160 bits. In addition, a message of any length may be encrypted, and the modular square root omitted. A variation to this scheme was proposed by Kurosawa and Desmedt [KD04]. Although the scheme by Kurosawa and Desmedt required fewer exponentiations, its assumptions required the use of a larger group (i.e. larger  $q$ ), making it of similar efficiency to Cramer and Shoup's earlier scheme. However, Gennaro and Shoup [GS04] have provided a new proof for this scheme, removing the assumptions requiring the large group size and enabling  $q$  to be 160 bits. We refer to this scheme as the GSKD scheme.

Table 1 shows the number of exponentiations and prime generations required for the Cramer–Shoup signature scheme and basic encryption scheme. The notation  $(x/y)$  indicates an exponentiation using an  $x$  bit modulus and a  $y$  bit exponent. The notation  $(x/y/\text{Dbl})$  is similar, except that it indicates a double

A	B
Known: Private key	Known: Verification key of A
‘A sent $m$ to B’	
$\sigma = \text{Sig}_A(m, B, N_B)$	$N_B \in_R \{0, 1\}^k$
$\xrightarrow{m}$	$\xrightarrow{m, N_B}$
$\xleftarrow{m, \sigma}$	Check $\sigma = \text{Sig}_A(m, B, N_B)$ ; if not then abort.
	‘B received $m$ from A’

Protocol 11: Signature-based MT-authenticator,  $\lambda_{\text{SIG}}$  [BCK98]

Operation	Signing	Verification
Online Exp.	$2 \times (512/160/\text{Dbl})$ $2 \times (512/512)$	$4 \times (1024/160)$
Online Prime Gen.	$1$ ( $\approx \frac{1}{3}$ total time)	
Online Time (ms)	50	57

Operation	Encryption	Decryption
Online Exp.	$1 \times (1024/1024)$	$[1] \times (1024/1024/\text{Dbl})$ $1 \times (1024/1024)$ $2 \times (1024/160)$
... for hybrid validation		$1$ ( $\geq 1 \times (1024/1022)$ exp.)
Online Modular Sqrt.		
Online Time (ms)	118 (estimated)	354 (estimated)
Online Hybrid Time	19 (16% of above)	57 (16% of 3 exp.)
Offline Exp.	$(3+[1]) \times (1024/1024)$	
Offline Time (ms)	472 (estimated)	
Offline Hybrid Time	57 (16% of 3 exp.)	

$[x]$  indicates  $x$  operations may be omitted in the GSKD hybrid scheme

Table 1: Efficiency of secure Signature and Encryption Schemes

exponentiation (the product of two exponentiations is required but both can be computed efficiently at the same time). Furthermore, if exponents for encryption and decryption are reduced to 160 bits instead of approximately 1024 bits, the number of exponentiations for the hybrid schemes can be ascertained. However, the hybrid scheme needs an additional 2 exponentiations to validate that the inputs to the decryption are members of the subgroup.

The signature scheme timings in the table are from Cramer and Shoup [CS00]; as they also give exponentiation timings, these have been used to estimate the time for the encryption schemes (the time required for symmetric operations in the hybrid scheme is ignored). Any optimization possible by using precomputation has been ignored in the estimations of time for the encryption scheme. In comparison with RSA, the signing algorithm is 2.3 times slower. This ignores the key setup phase which takes 31 milliseconds to build tables to speed up the main signing phase of 50 milliseconds. The efficiency of the GSKD hybrid encryption scheme appears to be a better choice for use with  $\lambda_{\text{P-ENC}}$  than the basic encryption scheme.

## 6 Comparison of Protocols

This section provides a comparison of the proposed, AP and GPAKE protocols. It assumes there is only one server (so the server’s identity or public key may be used in offline computations), the password may not be used in offline computations, and the time for hashes and MACs is generally ignored. Exponentiations use a 1024 bit modulus with a 160 bit exponent unless otherwise specified. If the shared secret DH value generated by using the key agreement protocol can be used directly as a key, then  $k$  in Assumption 1 would typically be 160 bits for 80-bit security. However, if a uniformly distributed key (e.g. a 128-bit key) is to be derived from the DH value,  $k$  would need to be larger unless a random oracle model is used to extract randomness from the DH value (e.g. using the left over hash lemma,  $k = 160 + 128 = 288$  bits for 79-bit security when the key is derived using a universal hash function). However, the exponents

Protocol	Client Efficiency			Server Efficiency			
	Operation	Equivalent Exp.		Operation	Equivalent Exp.		
		Offline	Online		Offline	Online	
Proposed	1 offline exp.	1		4 online PK dec.	12		
	1 online exp.	1					
	1 online PK enc.	3	1				
	1 offline PK enc.	4					
	Proposed Total:	8	2				Proposed Total:
GPAKE ...KOY	One-time signature key gen.			Per KOY key exchange:			
	One-time signature			One-time verification			
	Validity check	5		Validity check	5		
	2 offline exp.	2		3 offline exp.	3		
	2 double exp.	2.6		1 double exp.	1.3		
	2 multi-exp.	2	3.3	2 multi-exp.	2	3.3	
	KOY Total:	4	11	KOY Total per key:	5	9.6	
	...key dist.	1 symm. dec.			2 symm enc.		
	...other	2 exp.	1	1			
	GPAKE Total	5	12	GPAKE Total:	10	19.3	
AP	2 exp.	1	1	2 exp.	2		
	1 $G_1$ evaluation	4.6		2 $G_1$ evaluations	9.2		
	1 $G_2$ evaluation	4.6		2 $G_2$ evaluations	9.2		
	AP Total:	1	10.2	AP Total:	20.4		

Table 2: Protocol Efficiency Comparison

may remain at 160 bits if the 160-Discrete Log Short Exponent (DLSE) assumption is made. Further details are available in works by Dodis et al. [DGH<sup>+</sup>04] and Gennaro et al. [GKR04].

Abdalla and Pointcheval claim that the AP protocol only requires 2 exponentiations and a few multiplications per party. However, this figure does not take into account the  $G_1$  and  $G_2$  hash functions used by the protocol, which are modelled as random oracles in the proof, and map into the group,  $G$ , in which the protocol operates. The algebraic setting for the AP protocols is not specified, but a typical choice for  $G$  would be a subgroup of order  $q$  (where  $q$  is 160 to 288 bits) in  $\mathbb{Z}_p^*$  where  $p$  is a 1024 bit prime. How should  $G_1$  and  $G_2$  be implemented in such a setting? The most natural choice seems to be that used by MacKenzie [Mac02] for his suite of PAK protocols, and by IEEE P1363.2 [IEE06] in the DLREDP-1 (Discrete Log Random Element Derivation Primitive), in which the functions are implemented as a hash of the inputs raised to a power to map the hash output into the subgroup. This power will be quite large ( $1024 - 288 = 736$  bits long), and the computation would take the time of about 4.6 exponentiations with 160 bit exponents. Table 2 shows the equivalent number of exponentiations for this option. We should note, however, that a more efficient implementation may be possible when using DLREDP-2 from IEEE P1363.2 [IEE06] (but this requires adding two extra values to the domain parameters) or when  $\frac{p-1}{q}$  is small or  $G$  is an elliptic curve group (since the co-factor in these cases is much smaller).

The efficiency of GPAKE mainly depends on which 2-party password based key exchange protocol is used by it. Use of the KOY protocol [KOY01] is assumed here, since it is in the standard model and was suggested as a suitable protocol in the GPAKE proposal. (Other protocols recommended, although faster, were in the RO model.) The equivalent number of exponentiations for the KOY protocol, excluding those for the one-time signature, is shown in Table 2. (Some exponentiations may be performed more efficiently together, and we have accounted for this.) The Bellare and Rogaway [BR95] key distribution scheme was suggested for use with GPAKE, and requires 1 symmetric decryption for each client, and 2 symmetric encryptions for the server. The GPAKE protocol itself requires 2 exponentiations for each client.

The proposed protocol requires 1 offline and 1 online public key encryption, and 1 offline and 1 online exponentiation for each client, as well as 4 public key decryptions for the server.

From the summary in Table 2, we see that the GPAKE protocol is the least efficient of all the schemes, especially if offline computations are considered. Including  $G_1$  and  $G_2$  evaluations in the

efficiency analysis of AP makes its online efficiency similar to that of GPAKE, although it requires fewer offline computations. In comparison with these two schemes, our scheme performs quite favourably. However, some symmetric operations required for the public key encryption and decryption operations in the proposed scheme have been omitted from the analysis; symmetric operations have not been included in the GPAKE totals, either. In addition, it may be possible to optimize these schemes with precomputations to make the exponentiations run faster.

Another consideration in the choice of a protocol is the number of rounds it requires. Protocol 9 requires 6 rounds and Protocol 10 requires 3 rounds. In comparison, the GPAKE protocol requires 3 rounds for the KOY protocol, 3 rounds for the key distribution (although a small change to the key distribution protocol would result in 2 rounds), and 1 round for the exchange of the DH and MAC values, making a total of 6 or 7 rounds. The AP protocol requires only 2 rounds, but unlike the protocols described here, does not allow full concurrency. To allow full concurrency, an extra round must be added to the AP protocol, which would result in a protocol with the same number of rounds as Protocol 10.

In addition to the above efficiency considerations, there are a number of other points to consider when selecting a protocol:

**Random oracle v. standard model.** Our proof is in the standard model, whereas that of the AP protocol is in the RO model. The proof of GPAKE may be in either model, depending upon the components used with it.

**Corrupt queries.** The CK model, which is used for our proof, allows the use of adaptive Corrupt queries to model a malicious insiders. This is contrary to the AFP model. In fact, an attack on the AP protocol has been described [CBH05b] that uses a corrupt query. The original proof did not rule out this attack because corrupt queries were not allowed in the proof model. Therefore, we can rule out a wider range of attacks on our protocol because of the more general model.

**Session-state reveal queries.** The CK model also allows session-state reveal queries. These queries model the exposure of data that must be kept between activations, and the requirement that exposure of such data in one session should not compromise other sessions. However, the AFP model does not allow such queries, and the AP protocol would be insecure in the presence of such queries. This is because an adversary could find a party's secret Diffie-Hellman exponent using a session-state reveal query, and then use the exponent to find the secret password from the party's first protocol message.

**Key privacy.** The AFP model includes the key privacy requirement, but the CK model does not. However, the proposed protocol fulfills the requirement, since the server only forwards authentic messages in Protocol 1, and does not create any messages with new content. These messages are also known to the adversary, who cannot find any information about the key. Since the server knows no more than the adversary, it is also unable to deduce any information about the secret key. However, the CK model may need some modification to include the requirement if needed for other protocols.

**Concurrency.** The proof of the AP protocol does not allow concurrency, so that only one instance of a player can exist at a time. On the other hand, the proposed protocol allows full concurrency. Although instructions are given on how to modify the AP protocol to achieve partial and full concurrency, the correctness of these instructions is not proven. In addition, provision of full concurrency requires two extra message flows from the server at the beginning of the protocol.

**Assumptions.** The AP protocol proof requires the use of some non-standard assumptions based on the Diffie-Hellman problem. Our proposed protocol only requires the DDH assumption, a collision-resistant one-way hash function, and an IND-CCA secure encryption scheme. The use of standard assumptions may be less risky than the use of new assumptions that have not been extensively studied.

**Online attack detection.** It is always possible for the adversary to make an online attack by guessing a party's password and running the protocol using the guessed password. If the final key is correct, then the password must have been correct. Otherwise another password may be guessed and the protocol run again until the correct password is found. To prevent such attacks, the definition

of PBSK-security for the CK model requires the server to refuse to run the protocol with any party who has made more than a certain number of incorrect guesses of the password. However, the server in the AP and KOY protocols cannot detect whether a client has made an incorrect password guess, and although the proofs differentiate active and passive attacks, this cannot be done in practice. In addition, such attacks are tolerated by the AFP security definition where the problem is that the adversary's advantage must only be less than  $O(n/|\mathcal{D}|)$  plus a negligible function, where  $n$  is the number of active sessions and  $|\mathcal{D}|$  is the size of the password dictionary. No small bound is placed on the number of (unsuccessful) online attacks; it may be the same as the maximum number of online sessions.

In conclusion, we have proposed a new password-based three-party protocol, one of the few such protocols with a security proof. Our proof is in the standard model, in contrast to a recently proposed protocol with a proof in the RO model, and has a number of other advantages that have been discussed above.

## References

- [AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In *Public Key Cryptography—PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2005.
- [AP05] Michel Abdalla and David Pointcheval. Interactive Diffie-Hellman assumptions with applications to password-based authentication. In *Financial Cryptography and Data Security—FC 2005*, volume 3570 of *Lecture Notes in Computer Science*, pages 341–356. Springer, 2005. Full version: <http://www.di.ens.fr/~pointche/pub.php?reference=AbPo05>.
- [BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428. ACM Press, 1998. Full version at <http://www-cse.ucsd.edu/users/mihir/papers/key-distribution.html>.
- [Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.
- [BPR00] Mihir Bellare, David Pointcheval, and Phillip 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. Springer-Verlag, 2000.
- [BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1993. Full version at [www-cse.ucsd.edu/users/mihir](http://www-cse.ucsd.edu/users/mihir).
- [BR95] M. Bellare and P. Rogaway. Provably secure session key distribution – the three party case. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pages 57–66. ACM Press, 1995.
- [CBH05a] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Errors in computational complexity proofs for protocols. In *Advances in Cryptology—Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 624–643. Springer-Verlag, 2005.
- [CBH05b] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Examining indistinguishability-based proof models for key establishment protocols. In *Advances in Cryptology—Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 585–604. Springer, 2005.
- [CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer-Verlag, 2001. <http://eprint.iacr.org/2001/040.ps.gz>.

- [CN00] Jean-Sebastien Coron and David Naccache. Security analysis of the Gennaro-Halevi-Rabin signature scheme. In *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 91–101. Springer, 2000.
- [CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. International Association for Cryptologic Research, Springer, 1998.
- [CS00] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.
- [CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
- [DGH<sup>+</sup>04] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In *Advances in Cryptology — CRYPTO 2004 Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 494–510. Springer, 2004.
- [GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology—EUROCRYPT '99*, volume 1599 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, Berlin Germany, 1999.
- [GKR04] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Secure hashed Diffie-Hellman over non-DDH groups. In *Advances in Cryptology — EUROCRYPT 2004 Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 361–381. Springer, 2004. Full version in: Cryptology ePrint Archive (<http://eprint.iacr.org/2004/099>), Report 2004/099.
- [GS04] Rosario Gennaro and Victor Shoup. A note on an encryption scheme of kurosawa and desmedt. Cryptology ePrint Archive, Report 2004/194, 2004. <http://eprint.iacr.org/2004/194>.
- [HBGN05] Yvonne Hitchcock, Colin Boyd, and Juan Manuel González Nieto. Modular proofs for key exchange: rigorous optimizations in the Canetti-Krawczyk model. *Applicable Algebra in Engineering, Communication and Computing (AAECC) Journal*, 2005. Special issue on Mathematical Techniques in Cryptology; <http://dx.doi.org/10.1007/s00200-005-0185-9>.
- [HK99] Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and Systems Security*, 2(3):230–268, 1999.
- [HTB<sup>+</sup>03] Yvonne Hitchcock, Yiu Shing Terry Tin, Colin Boyd, Juan Manuel González Nieto, and Paul Montague. A password-based authenticator: Security proof and applications. In *4th International Conference on Cryptology in India – INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.
- [IEE06] IEEE (Institute of Electrical and Electronics Engineers, Inc.). P1363.2: Standard specifications for password-based public-key cryptographic techniques (draft version d23), 2006. <http://grouper.ieee.org/groups/1363/passwdPK/draft.html>.
- [KD04] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004.
- [KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, 2001.

- [Mac02] Philip MacKenzie. The PAK suite: Protocols for password-authenticated key exchange. Technical Report 2002-46, DIMACS, 2002. [Online] <ftp://dimacs.rutgers.edu/pub/dimacs/TechnicalReports/TechReports/2002/2%002-46.ps.gz> [accessed 14/05/2003].
- [NPS01] David Naccache, David Pointcheval, and Jacques Stern. Twin signatures: an alternative to the hash-and-sign paradigm. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 20–27, Philadelphia, PA, USA, nov 2001. ACM Press.
- [Sho00] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT '2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer, 2000.