A Provably Secure One-Pass Two-Party Key Establishment Protocol

K. Chalkias, S.T. Halkidis, D. Hristu-Varsakelis, G. Stephanides, and A. Alexiadis

Computational Systems and Software Engineering Laboratory,
Department of Applied Informatics,
University of Macedonia, Thessaloniki, Greece
{chalkias,halkidis}@java.uom.gr,{dcv,steph}@uom.gr,talex@java.uom.gr

Abstract. For two parties to communicate securely over an insecure channel, they must be able to authenticate one another and establish a common session key. We propose a new secure one-pass authenticated key establishment protocol which is well suited to one-way communication channels. The protocol is examined using an extension of the Bellare-Rogaway model proposed by Blake-Wilson et. al., and is shown to be provably secure, in the sense that defeating the protocol is equivalent to solving a CDH problem. We compare our protocol to existing approaches, in terms of security and efficiency. To the best of our knowledge, ours is the only one-pass protocol that resists general key-compromise impersonation attacks, and avoids certain vulnerabilities to loss of information attacks found in other protocols of its class.

Keywords: One-pass protocols, two-party key agreement, key-compromise impersonation, loss of information.

1 Introduction

In the last few years, there has been increasing interest in secure two-party key agreement protocols, in large part because of the need for protecting communications over public, unreliable channels. In that context, the protection and authenticity of the messages to be exchanged hinges on the establishment of a group symmetric session key. The pioneering work in two-party key establishment was the Diffie-Hellman protocol [14], which nevertheless suffered from several security problems, such as vulnerability to man-in-the-middle attacks. Efforts to improve on the early Diffie-Hellman protocol have given rise to various non-ID based authenticated two-party key agreement protocols, including recently proposed one round, [18,24], two round [6,25] and three round protocols [8,10,22]. A complementary approach has focused on ID-based schemes [27,13,31,30], which achieve their main security goals but may be quite slow because of their extensive use of bilinear pairings [32]. A common disadvantage of the protocols mentioned here is that they impose either a high computational cost or high communication cost in order to provide authentication. Moreover, their majority requires at least two "rounds", making them unsuitable for one-way communication channels (e.g., e-mail, sms, store-and-forward applications).

This paper proposes a new one-pass two-party key establishment protocol that rectifies some of the problems associated with existing one-pass approaches. The proposed scheme is a slight adaptation on the basic elliptic curve Diffie-Hellman (ECDH) protocol, equipped with an authentication mechanism based on bilinear pairings. To establish a session key, the sender (the initiator of the one-way protocol) generates an ephemeral key-pair and sends its public part to the receiver. To achieve authentication, a similar technique to the short signature scheme proposed in [9] is used. Finally, time-stamp and identities are used to tie quantities to particular entities and reduce the replay vulnerability.

Our approach is robust to unknown key-share (UK-S) attacks and achieves the highest possible level of security against key-compromise impersonation (K-CI) attacks for one-pass protocols, where an attacker who somehow learns a user's private key can impersonate any other entity to the victim, potentially gaining much more knowledge than by simply having access to the victim's past or future conversations. We will have more to say about this in Section 5.4. Furthermore, our protocol is not affected by a loss of information (LoI) attack which can be mounted against other one-pass approaches.

The remainder of this paper is organized as follows: In Section 2 we fix notation and review some required definitions. Section 3 describes the proposed protocol. A security analysis is presented in Section 4. We discuss various desirable attributes of our protocol in Section 5. Section 6 makes comparisons with other widely used schemes.

2 Preliminaries

For the purposes of this work, we will require an abelian, additive finite group $\mathbb{G}_1$, of prime order q, and an abelian multiplicative group, $\mathbb{G}_2$, of the same order. For example, $\mathbb{G}_1$ may be the group of points on an elliptic curve. We will let P denote the generator of $\mathbb{G}_1$. Also, H_1, H_2, will be two secure hash functions, with $H_1 : \{0,1\}^* \mapsto \mathbb{G}_1$ and $H_2 : \{0,1\}^* \mapsto \{0,1\}^k$, where $k \in \mathbb{Z}_+^*$. We will write $a \in_R S$ to denote an element a chosen at random from S. Finally, $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ will be a bilinear pairing, defined below.

Definition 1. *Let $\mathbb{G}_1$ be an additive cyclic group of prime order q generated by P, and $\mathbb{G}_2$ be a multiplicative cyclic group of the same order. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ is called a bilinear pairing if it satisfies the following properties:*

- *Bilinearity: $\hat{e}(aV, bQ) = \hat{e}(abV, Q) = \hat{e}(V, abQ) = \hat{e}(V, Q)^{ab}$ for all $V, Q \in \mathbb{G}_1$ and $a, b \in Z_q^*$.*
- *Non-degeneracy: there exist $V, Q \in \mathbb{G}_1$ such that $\hat{e}(V, Q) \neq 1$.*
- *Efficiency: there exists an efficient algorithm to compute the bilinear map.*

Admissible bilinear pairings can be constructed via the Weil and Tate pairings [12,32]. For a detailed description of pairings and conditions under which they can be applied to elliptic curve cryptography, see [12,32].

3 Proposed Protocol

In this Section we describe a new one-pass two-party authentication key establishment protocol. It is composed of two phases, *protocol initialization* and *protocol running*, described next.

3.1 Protocol Initialization

Consider two players, A and B, which are to establish a common session key. First, A chooses a random $x_A \in \mathbb{Z}_q^*$ as her private key, and computes $Y_A = x_A P$ to be her corresponding public key. Similarly, B chooses a random $x_B \in \mathbb{Z}_q^*$ as his private key, and computes his public key $Y_B = x_B P$. We will let the strings ID_A, ID_B denote the identities of A and B, respectively.

3.2 Protocol Running

To establish a session key, A and B obtain the public key of one another and execute the protocol shown in Table 1.

Table 1. Proposed two-party authenticated key agreement protocol

A $(x_A, Y_A = x_A P)$		**B** $(x_B, Y_B = x_B P)$
$\alpha \xleftarrow{R} \mathbb{Z}_q^*, \quad X_1 = \alpha P$		
$X_2 = \alpha Y_B = \alpha x_B P$		
$Q = H_1(X_2 \| ID_A \| ID_B \| T_1)$		
$Y_1 = x_A Q$		
$sk = H_2(X_2 \| ID_A \| ID_B \| T_1)$	$\xrightarrow{(X_1, Y_1, T_1, ID_A)}$	$X_2 = x_B X_1 = \alpha x_B P$
		$Q = H_1(X_2 \| ID_A \| ID_B \| T_1)$
		$\hat{e}(Y_A, Q) \overset{?}{=} \hat{e}(P, Y_1)$
		$sk = H_2(X_2 \| ID_A \| ID_B \| T_1)$

1. Player A selects a random number $\alpha \in \mathbb{Z}_q^*$, and computes: $X_1 = \alpha P$, $X_2 = \alpha Y_B$, $Q = H_1(X_2||ID_A||ID_B||T_1)$, $Y_1 = x_A Q$, and the session key $sk = H_2(X_2||ID_A||ID_B||T_1)$, where T_1 is a time-stamp and the symbol $||$ denotes string[1] concatenation. Then, A sends X_1, Y_1, T_1 and its identity, ID_A, to B.

[1] When elements of a group appear as arguments of a hash function (e.g., $H_1(X_2||ID_A||ID_B||T_1)$), it will be understood that string representations of these elements are used.

2. Upon receipt of (X_1, Y_1, T_1, ID_A), player B computes $X_2 = x_B X_1 = \alpha Y_B$, $Q = H_1(X_2||ID_A||ID_B||T_1)$ and checks whether $\hat{e}(Y_A, Q) \overset{?}{=} \hat{e}(P, Y_1)$. If equality does not hold, B terminates the protocol. Otherwise, he computes the session key $sk = H_2(X_2||ID_A||ID_B||T_1)$.

Correctness: In an honest execution of the protocol A and B share the common key $sk = H_2(X_2||ID_A||ID_B||T_1)$.

Efficiency: Our *one-pass* protocol requires four integer-to-point multiplications, two map-to-point hashes, two plain hashes and two pairings. The computational cost of pairings will be discussed further in Section 6.3.

4 Provable Security

Although the concept of provable security has occasionally taken some criticism, it remains one of the few formal approaches used to make precise statements regarding the security of protocols, and therefore continues to be widely used. In this work, we prove the security of our protocol in an environment which is an extension of the Bellare-Rogaway model [5] proposed by Blake-Wilson et. al. [7]. Our choice of [7] is partly motivated by the fact that in our protocol, authentication and key generation are intertwined (for example, X_2 is used to generate both Q and the secret key). In particular, this means that the protocol is not amenable to the "modular" approach of Canneti-Krawzcyk [11] where a protocol is first proved secure in the absence of authentication and then an appropriate authentication layer is added.

4.1 The Computational Diffie-Hellman Problem

The security of our protocol will turn out to be linked to the well-known Computational Diffie-Hellman (CDH) problem. The CDH problem in $\mathbb{G}_1$ [14], is to compute $\alpha\beta P$ given $P, \alpha P$, and βP, for some fixed $\alpha, \beta \in \mathbb{Z}_q^*$. It is considered computationally hard, assuming an adversary that runs in polynomial time. In our protocol, we have precisely an instance of the CDH problem, because the adversary must compute $\alpha x_B P$ given αP (transmitted by A) and $x_B P$ (B's public key), in order to find the session key.

4.2 Security Analysis

We proceed to show that the proposed protocol is secure in the random oracle model [5,7], assuming that the CDH problem is hard in $\mathbb{G}_1$. We first give a brief, intuitive description of the main assumptions regarding the environment in which the protocol is run. Additional details can be found in [7].

We consider a collection of entities-players which may communicate with one another by initiating the protocol. An adversary can eavesdrop on communications, reroute or change the content of messages, initiate sessions between entities, engage in multiple sessions with the same entity at the same time, and ask an entity to enter a session with itself. The adversary is a probabilistic polynomial-time Turing Machine and has access to a collection of oracles $\{\Pi^s_{i,j}\}$, where $\Pi^s_{i,j}$ behaves as entity i carrying out a protocol session in the belief that it is running the protocol with entity j for the s-th time. We will assume that $1 \leq s \leq \tau(k)$, where τ is polynomial in the security parameter k of the protocol.

The adversary may choose to give an oracle a message (of the form $(X_1, Y_1, ID_A, ID_B, T)$) as an input, via a `Send` query. A `Reveal` query tells an oracle to reveal its current session key. An oracle which has been asked a `Reveal` query is said to be *opened*. Finally, a `Corrupt` query, targeted at entity i, tells all $\Pi^s_{i,j}$ oracles to reveal i's long-term private key to the adversary and to replace i's key pair with any valid key pair chosen by the adversary. An oracle $\Pi^s_{i,j}$ for which a `Corrupt` query has been asked for i, is said to be *corrupted*. We say informally that an oracle has *accepted*, if it has successfully terminated a protocol run. An oracle Π^s_{ij} is *fresh* if it has accepted, i and j are both uncorrupted, it is unopened, and there is no opened oracle $\Pi^t_{j,i}$ with which it has had a *matching conversation*[2], as defined in [5,7].

In the setting described above, the adversary begins by asking the oracles all the queries it wishes to make. He then selects any fresh oracle $\Pi^s_{i,j}$ and asks a single new query, called `Test`. To answer the query, the oracle flips a fair coin $b \overset{R}{\leftarrow} \{0, 1\}$ and returns the session key if $b = 0$ or a random value if $b = 1$. The adversary "wins" at this experiment if he correctly guesses the value of b.

Definition 2. *A protocol P is a secure One-Pass Authenticated Key establishment (OPAK) protocol if:*

1. *In the presence of a benign adversary on $\Pi^s_{i,j}$ and $\Pi^t_{i,j}$, (i.e., an adversary that does not interfere with the routing or contents of messages) both oracles always accept holding the same session key κ, and this key is distributed uniformly at random on $\{0, 1\}^k$, where k is the protocol's security parameter.*
2. *If uncorrupted oracles $\Pi^s_{i,j}$ and $\Pi^t_{i,j}$ have matching conversations then both oracles accept and hold the same session key κ.*
3. *The adversary makes no* `Reveal` *queries.*
4. *Let $GoodGuess^E(k)$ be the probability that the adversary, E, correctly guesses the coin flip at the* `Test` *query. Then,*
   $advantage^E(k) = \left|Pr[GoodGuess^E(k)] - \frac{1}{2}\right|$ is negligible[3].

In the following we will show that the protocol described in Table 1 is a secure OPAK protocol.

[2] Intuitively, two oracles are said to have matching conversations if one of them is the initiator of an exchange of messages, and the messages sent by each of the two are identical to those received by the other, and are in the same temporal order. See [5,7] for a precise definition. In our case, there is only one message $msg = (X_1, Y_1, T_1, ID_A)$ transmitted during the protocol, so that matching conversations have a particularly simple form: the initiator oracle takes as input the empty string, λ, and transmits msg; the responder takes msg as input and transmits λ.

[3] A real-valued function $\epsilon(k)$ is called negligible if for every $c > 0$ there exists a $k_c > 0$ such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

Remark 1. We note that Def. 2 is a modified version of Def. 10 in [7]. Our condition 3 is necessary because all one-pass protocols are prone to replay attacks, thus the strongest property of Authenticated Key (AK) security [7] cannot be achieved. If `Reveal` queries were allowed then an adversary E could win the `Test` by a replay attack. E could submit the same properly formatted message to two receiver instances (formally, E initiates oracles Π^s_{AB}, Π^u_{AB}, corresponding to two distinct sessions between A and B). Both receiver instances will accept and produce the same output session key. Then, E `Reveals` the key of one of the sessions, `Tests` against the other session, and wins.

Remark 2. If one insists on keeping `Reveal` queries in play, then the weakness with respect to replay attacks can be somewhat rectified in practical settings if each entity maintains a list of secret keys used during the latest time period. The length of that list would depend on the frequency with which T_1 is updated. An entity would be alerted to a replay attack if a session key it generates is present in the list. Session keys from previous time periods need not be memorized because they could not be successfully replayed to a received.

Remark 3. The notion of OPAK security could actually be strengthened slightly, by allowing the adversary to make `Reveal` queries, but stipulating that if such a query is issued to an oracle Π^s_{AB}, all oracles Π_{AB} are marked as opened.

Theorem 1. *The protocol shown in Table 1 is a secure OPAK protocol, provided that the CDH problem is computationally hard and H_1, H_2 are independent random oracles.*
Proof: See Appendix.

5 Protocol Attributes

In the following we discuss a series of security attributes as they pertain to our protocol. We will forgo formal proofs where applicable because of space limitations.

5.1 Known Session-Keys

Known session-key security [7], also known as known-key security (K-KS), means that a protocol still achieves its goal in the face of an adversary who has learned some previous session keys. One-pass protocols without time-stamps cannot achieve K-KS since an adversary can simply replay the information from a previous protocol run. time-stamps allow a one-pass protocol to achieve some measure of K-KS, meaning that they reduce but do not eliminate the vulnerability to attacks. More specifically, entity B can check the time-stamp T_1 sent by A, and terminate the protocol if too much time has elapsed since then. Of course, this requires synchronization of A's and B's clocks, to within some reasonable tolerance. Depending on the transmission delay imposed by the communication channel, an entity can set a time threshold that leaves a potential attacker little time to mount a known session key attack. If A's and B's clocks are perfectly synchronized and the transmission delay is known with certainty, then the time left for an attack could be made arbitrarily small. The question of what is an acceptable time threshold will generally be application-dependent, and will not be discussed further here.

5.2 Forward Secrecy

Perfect forward secrecy (PFS) means that if long term secrets of one or more entities are compromised, the secrecy of previously computed session keys is not affected. Our protocol does not achieve PFS. To see this, note that if the adversary learns the secret value of B, x_B, then knowing $X_1 = \alpha P$, which is transmitted in the clear, the adversary can compute $X_2 = x_B \alpha P$. Because timestamps are also transmitted in the clear, the adversary can also compute the session key $sk = H_2(X_2||ID_A||ID_B||T_1)$ of a previous session. This is not surprising in light of the fact that there exists no protocol for implicit authentication that achieves PFS with two or fewer messages [21]. However, similarly to the majority of one-pass approaches [21,24], our protocol does achieve partial forward secrecy, because if the adversary learns the secret value of A, x_A, he still faces the CDH problem of computing $\alpha x_B P$ from αP and $x_B P$, before finding any previous session key.

5.3 Unknown Key-Share

Prevention of unknown key-shares (UK-S) as defined in [7], is a property held by all AK-secure protocols. In that setting, an entity i is coerced into sharing a key κ with entity f, while entity j is coerced into holding κ in the belief that it is shared with i. Our OPAK protocol is also secure against UK-S of that type, as well as against the more typical (and broader) UK-S attack [20,28,24,4], where f is not required to possess the key held by i and j, but merely to confuse j as to the identity of i, with whom j shares a key. In our case, UK-S is prevented via the inclusion of the parties' identities in the computation of the session key. A formal proof can easily be constructed around the following basic argument. In order to accomplish a UK-S, the adversary must at least alter the ID information transmitted by i to j. Even if the altered data are such that j's bilinear pairing test is successful, the key computed by j will not be the same as that held by i, thus no key sharing is accomplished. This technique seems to be a general mechanism for preventing UK-S attacks [23].

5.4 Key-Compromise Impersonation

Resistance to key-compromise impersonation (K-CI) attacks means that if i's secret value is disclosed to an adversary, then the adversary cannot impersonate other entities to i [7]. We stress the importance of a protocol being secure against impersonation threats, as an attacker of this type can feign a trusted entity to the victim, and thus ask for (and receive) important information[4].

In fact, there is a special K-CI attack that apparently succeeds with all one-flow protocols. An intruder C that learns B's secret key and then eavesdrops one message from A, (X_1, Y_1, T_1, ID_A), would be able to impersonate A (but *no one else*) to B and only for the current session, because C is also able to create the current session key. However, this attack is more limited than the general K-CI attack, in which the intruder can impersonate *any* entity and at *any* time, to which other one-pass schemes are open (see comparisons in Sec. 6.2).

In our case, if an adversary, C, obtains the secret value of B, x_B, he cannot impersonate another entity, J, to B (excluding the special case described in the previous paragraph); in that sense, our protocol achieves a reduction of the K-CI vulnerability to the greatest extent possible for one-pass approaches. To see why that is, note that C must "pass" the bilinear pairing test, $\hat{e}(Y_J, Q) = \hat{e}(P, Y_1)$, performed by B to successfully impersonate J. If C knows B's private key, x_B, he can initially create a X_1 value from which he can compute Q. C is then faced with the task of computing an appropriate Y_1 such that $\hat{e}(Y_J, Q) = \hat{e}(P, Y_1)$. One can easily show that if C can do this, then with high probability he can compute $x_J Q$ from Q and $x_J P$. By rewriting $Q = kP$ for some unknown k, the last statement is equivalent to C finding (with high probability) $x_J kP$ from kP and $x_J P$, which is an instance of the CDH problem.

Finally, we note that there is no need to examine the case where an adversary obtains the secret value x_A of A and impersonates B to A, since B does not reply to A. In that sence, all of the modern one-pass approaches, including ours, achieve K-CI resilience in the initiator's side [8].

5.5 Loss of Information

Loss of the value X_2 (or loss of Q) from a previous protocol run does not affect the security of subsequent protocol runs, because the computation of X_2 is based on the random value α. Usually a loss-of-information attack succeeds when authentication is used with values which are not random. Some examples are the protocols proposed in [19] and [18], where authentication is based on the computation of $x_A x_B P$ [5]. In contrast to the majority of one-pass key establishment schemes [21,24,1,26,15,29], we do not make use (directly or indirectly) of $x_A x_B P$ values in our approach. We will return to the implications of this in Sec. 6.2. Finally, note that the private key of the receiver (B) is as secure as can be, because B transmits no information during the protocol run.

[4] If a private key is compromised, the attacker is able to intercept messages by eavesdropping on past or future conversations (e.g., e-mails). However, if a communication protocol is vulnerable to K-CI, the attacker would also be able to elicit additional information that may never have been communicated otherwise.

[5] This quantity is widely used in a number of cryptographic protocols; to date, the consequences of its exposure have not been adequately studied in the literature.

5.6 No Key Control and Message Independence

A protocol is said to have the “No Key Control” property if no participant (or adversary) can either control or predict the value of the session key. This property cannot be provided by any one-flow protocol, due to the fact that only the initiator of the protocol produces a random value, and so he can a-priori compute the session key (the receiver does not reply at all). In order to avoid key control, two or more passes are required.

A protocol is said to achieve message independence if individual flows of a protocol run between two honest entities are unrelated [7]. Because in our OPAK protocol only one flow exists, message independence is achieved. This is in contrast to AKC (AK with Key confirmation) protocols [7].

6 Comparison with Existing One-Pass Schemes

This section compares the security and efficiency of the proposed protocol against those of existing one-pass protocols.

6.1 Existing One-Pass Key Establishment Protocols

Because of the need for security in applications where only one entity is on-line, (e.g., secure e-mail, sms, store-and-forward), there have been numerous attempts at designing a secure and efficient one-pass key agreement protocol. The majority of existing one-pass schemes were created as variants of previously-proposed two-way or even two-round key agreement schemes. Some of the best known and most successful approaches to one-pass key establishment protocols include:

– The one-pass variant of the Key Exchange Algorithm (KEA) designed by the NSA and declassified in 1998 [29]. KEA is the key agreement protocol in the FORTEZZA suite of cryptographic algorithms designed by NSA in 1994 and it is similar to the Goss [15] and MTI/A0 [26] protocols.
– The one-pass variant of the Unified Model, proposed by Ankney, Johnson and Matyas [1]; it is an AK protocol that is in the draft standards ANSI X9.42 [2], ANSI X9.63 [3], and IEEE P1363 [17].
– The one-pass variant of the MQV protocol [24] that is in the draft standards ANSI X9.42 [2], ANSI X9.63 [3], and IEEE P1363 [17].
– The one-pass HMQV protocol [21], a variant of MQV.

6.2 Comparison in Terms of Security

Table 2 compares the proposed protocol to those listed in Section 6.1, in terms of various classes of attacks and security-related attributes.

Being one-pass, the protocols listed cannot provide PFS or No Key Control; also, none of the existing one-pass protocols provide K-KS, due to the possibility of replay attacks. The use of time-stamps and the independence of session keys

Table 2. Comparison of Security Properties. IKA denotes implicit key authentication, PFS perfect forward secrecy, K-KS known-key security, UK-S unknown key-share, K-CI key-compromise impersonation, and LoI loss of information $(x_A x_B P)$. $\surd$ denotes assurance; $\surd ?$ indicates that the assurance is provided modulo a technicality; $\surd^{-}$ denotes assurance, excluding the vulnerability to eavesdropping attacks discussed in Sec. 5.4; $\times^{+}$ indicates that the assurance is not provided by the protocol, unless modifications are made; $\times$ indicates that the assurance is not provided by the protocol.

	IKA	PFS	K-KS	UK-S	K-CI	LoI
KEA	$\surd$	$\times$	$\surd ?$	$\times^{+}$	$\times$	$\times$
Unified Model	$\surd$	$\times$	$\surd ?$	$\surd$	$\times$	$\times$
MQV	$\surd$	$\times$	$\surd ?$	$\times^{+}$	$\times$	$\times$
HMQV	$\surd$	$\times$	$\surd ?$	$\times^{+}$	$\times$	$\times$
Proposed OPAK	$\surd$	$\times$	$\surd ?$	$\surd$	$\surd^{-}$	$\surd$

help reduce (but not eliminate) the vulnerability to known-key attacks. In Table 2 we have used the symbol $\surd ?$ to indicate the fact that K-KS security should be regarded "modulo" the technicalities discussed in Section 5.1, namely clock synchronization and the time threshold used to decide whether the message A sends to B is "too old". We have marked all protocols listed with $\surd ?$ under K-KS because we assume that time-stamps can be added to any of them, just as we have done for ours.

Regarding security against UK-S attacks, [20] has presented an on-line UK-S attack on the MQV protocol, and [28] described a UK-S attack on the one-pass HMQV protocol. The KEA one-pass protocol is also prone to UK-S attacks as was shown in [23]. Of course, these protocols can be modified by incorporating parties identities in the computation of a session key, and thus become secure against UK-S threats[6]. This is the reason we have marked KEA, MQV and HMQV with $\times^{+}$ under UK-S. Unlike these protocols, there are no known UK-S attacks either on the Unified Model or on the protocol proposed here (see also the discussion in Section 5.3).

Of the protocols shown in Table 2, ours provides the highest level of security against K-CI attacks. This is because the bilinear pairing verification works as a short digital signature [9] on the secret key for the specified time period whereas, none of the previous approaches includes a sender verification mechanism. For instance, an exponential challenge-response (XCR) signature (from a player A to a player B), used in the HMQV protocol [21], can also be constructed by anyone who has knowledge of the recipient's private key. The latter means, that if an attacker has knowledge of B's private key, he is able to create a signature of this type and impersonate A to B. At this point, there do not seem to be any obvious modifications that would eliminate the K-CI vulnerability in the first four protocols shown in Table 2. We have marked our protocol with $\surd^{-}$ under

[6] In some cases (e.g., HMQV), the incorporation of parties' identities is already applied in some of the intermediate steps of the protocol; following our suggestion regarding the computation of the session key then leads to a significantly different protocol.

K-CI because no one-flow scheme can provide perfect K-CI security (for the reasons mentioned in Sec. 5.4). However, our protocol minimizes the vulnerability to such attacks: the class of entities which an adversary in possession of B's private key can impersonate is reduced from any entity / any time (with the KEA, MQV, HMQV and Unified Model protocols) to only an entity that attempts to communicate with B (and only for that session).

Finally, considering the possible loss of the static Diffie-Hellman exponent, $x_A x_B P$, our approach remains secure, as explained in Section 5.5. However, the security of the rest of the protocols examined here depends on the secrecy of this value. In particular, if $x_A x_B P$ were known to an adversary, an impersonation attack can easily take place. For instance, in the one-pass HMQV protocol, an adversary, say E, knowing[7] $L = x_A x_B P$, can initiate a new session with B by impersonating A to him. Using the same notation for the private/public keys as in our protocol, a session key between A and B, $sk = (\alpha + x_A d)Y_B$ (where $d = \bar{H}(X_1, ID_A, ID_B)$ [21]), could also be computed by E as $sk = \alpha Y_B + dL$. The same attack can be mounted against the Unified Model, KEA and MQV one-pass protocols.

6.3 Comparison in Terms of Computational Efficiency

Among the protocols examined, MQV and HMQV are the most efficient. They require two scalar multiplications for the initiator (A) and two for the receiver (B). For the KEA and Unified Model protocols, the corresponding figures are three and two, respectively. Under our protocol, the computational cost for the initiator is a bit higher (three scalar multiplications and one map-to-point hash)[8]. The recipient performs one scalar multiplication in order to create the session key, but must also compute one map-to-point hash and two bilinear pairings in order to verify the sender. Based on the estimates in [16,33], the computational cost of one bilinear pairing is approximately equal to that of four scalar multiplications in the abelian group $\mathbb{G}_1$when using a subgroup of order q in a supersingular elliptic curve E over F_p, where p is a 512 bit prime and q is a 160 bit prime. The pairing computation is the most expensive part of our protocol, but this is the price to be paid for improved security. We emphasize, however, that the reason for choosing the pairing-based short signature (BLS) scheme of [9] is that its signature length is half the size of a DSA signature for a similar level of security and thus the lowest communication cost is achieved.[9]. Because of this, and because pairings are only computed by the recipient, our protocol

[7] this would require E to compromise an earlier session key of B; prior to that, it is possible for E to send a properly formatted message to B such that B's session key is a known multiple of L.

[8] We ignored operations whose cost is negligible compared to that of a scalar multiplication in $\mathbb{G}_1$. These include generating random numbers, integer multiplication, plain hashes and point additions in $\mathbb{G}_1$.

[9] It is a fact that in cases where computational complexity on the receiver's side is of much more importance, one could replace BLS signatures with a more efficient scheme based on discrete logs.

is best-suited to systems where initiators use mobile or low-power devices, as well as in cases where data is sent over a low-bandwidth channel. For example, in smartcard transactions, the initiator of the protocol (e.g., hardware on the smartcard) will not have to compute any pairings; that cost will be borne by the typically more powerful CPU attached to the card reader.

7 Conclusions

This work proposed a new OPAK protocol that addresses the major security problems existing in protocols of its category. We have used the extension to the Bellare-Rogaway model [5] by Blake-Wilson et al. [7] to prove the security of the protocol. We have compared the proposed protocol to existing one-pass schemes, including the MQV (proposed by NSA as the standard key exchange protocol for the US government) and HMQV protocols, and discussed its strengths in several aspects of security. Our protocol achieves the highest level of security possible with one-pass approaches against all widely studied security attacks/properties (IKA, K-KS, UK-S, K-CI, partial forward secrecy and LoI), at the expense of slightly higher computational cost.

The proposed protocol is well-suited to applications with one-way communication channels. Examples include e-mail or sms, where the receiver cannot immediately reply, and store-and-forward applications (e.g., printers) where messages are sent to resources which need not reply at all. The low processing requirements for the protocol's initiator make our approach ideal for use in very low-end computing systems such as smartcards or high-load servers, because the sender does not have to compute any pairings, which are the most costly part of the protocol. Opportunities for further work include the conversion of our protocol to provide security in the standard model, perhaps using [18] as a basis.

References

1. Ankney, R., Johnson, D., Matyas, M.: The Unified Model, Contribution to X9F1 (1995)
2. ANSI X9.42, Agreement of Symmetric Algorithm Keys Using Diffie-Hellman, Working Draft (1998)
3. ANSI X9.63, Elliptic Curve Key Agreement and Key Transport Protocols, Working Draft (1998)
4. Baek, J., Kim, K.: Remarks on the Unknown Key-Share Attacks. IEICE Transactions on Fundamentals E83-A(12) (2000)
5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 110–125. Springer, Heidelberg (1994)
6. Bird, R., Gopal, I., Herzberg, A., Janson, P., Kutten, S., Molva, R., Yung, M.: Systematic Design of Two-Party Authentication Protocols. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 44–61. Springer, Heidelberg (1992)
7. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997)

8. Blake-Wilson, S., Menezes, A.: Authenticated Diffie-Hellman key agreement protocols. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 339–361. Springer, Heidelberg (1999)
9. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319 (2004)
10. Boyd, C., Mao, W., Paterson, K.G.: Key Agreement Using Statically Keyed Authenticators. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 248–262. Springer, Heidelberg (2004)
11. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
12. Cha, J., Cheon, J.: An Id-Based Signature from Gap-Diffie-Hellman Groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, Springer, Heidelberg (2002)
13. Chen, L., Kudla, C.: Identity Based Authenticated Key Agreement Protocols from Pairings. In: Proc. 16th IEEE Computer Security Foundations Workshop - CSFW 2003 (2003)
14. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
15. Goss, K.C.: Cryptographic Method and Apparatus for Public Key Exchange with Authentication, U.S. Patent 4956865 (1990)
16. Harrisson, K., Page, D., Smart, N.P.: Software Implementation of Finite Fields of Characteristic Three for Use in Pairing Based Cryptosystems. LMS Journal of Computer Mathematics 5(1), 181–193 (2002)
17. IEEE 1363, Standard Specifications for Public Key Cryptography, Working Draft (1998)
18. Jeong, I.R., Katz, J., Lee, D.H.: One-Round Protocols for Two-Party Authenticated Key Exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 220–232. Springer, Heidelberg (2004)
19. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
20. Kaliski Jr., B.S.: An Unknown Key-Share Attack on the MQV Key Agreement Protocol. ACM Transactions on Information and Systems Security 4(3), 275–288 (2001)
21. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
22. Kwon, T.: Authentication and Key Agreement via Memorable Password. In: NDSS 2001 Symposium Conference Proc. (2001)
23. Lauter, K., Mityagin, A.: Security Analysis of KEA Authenticated Key Exchange Protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
24. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography 28(2), 119–134 (2003)
25. Lu, R., Cao, Z., Su, R., Shao, J.: Pairing-Based Two-Party Authenticated Key Agreement Protocol, `http://eprint.iacr.org/2005/354.pdf`
26. Matsumoto, T., Takashima, Y., Imai, H.: On Seeking Smart Public-Key Distribution Systems. Transactions of the IECE of Japan E69, 99–106 (1986)

27. McCullagh, N., Barreto, P.S.L.M.: A New Two-Party Identity-Based Authenticated Key Agreement. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 262–274. Springer, Heidelberg (2005)
28. Menezes, A.: Another look at HMQV, IACR Eprint archive (2005), `http://eprint.iacr.org/2005/205`
29. NIST, SKIPJACK and KEA Algorithm Specification (1998), `http://csrc.nist.gov/encryption/skipjack/skipjack.pdf`
30. Scott, M.: Authenticated ID-based Key Exchange and remote log-in with simple token and PIN Number (2002), `http://eprint.iacr.org/2002/164`
31. Smart, N.P.: An Identity Based Authenticated Key Agreement Protocol based on the Weil Pairing. Electronics Letters 38(13), 630–632 (2002)
32. Stögbauer, M.: Efficient Algorithms for Pairing-Based Cryptosystems, Diploma Thesis: Department of Mathematics, Darmstadt University of Technology (2004)
33. Yoon, H.J., Cheon, J.H., Kim, Y.: Batch verifications with ID-based signatures. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 233–248. Springer, Heidelberg (2005)

Appendix: Proof of Theorem 1

Conditions 1 and 2: From the protocol definition it is clear that, in the presence of a benign adversary on Π_{ij}^s and Π_{ji}^t, as well as when uncorrupted oracles Π_{ij}^s and Π_{ji}^t have matching conversations, both oracles accept and hold the same session key $sk = H_2(ax_jP \| ID_i \| ID_j \| T_1)$.
Condition 3: Consider an adversary, E. We will show that if $advantage^E(k)$ is non-negligible we are led to a contradiction, and in particular that it is then possible to construct a new adversary, F, which succeeds in solving the CDH problem with non-negligible probability.

Suppose that E chooses some fresh oracle, $\Pi_{i,j}$, to ask its `Test` query. The key held by such an oracle will be of the form $H_2(ax_jP \| ID_i \| ID_j \| T_1)$. Let Γ_k be the event that H_2 has previously been queried on an input that begins with ax_jP, by E or by some oracle other than a $\Pi_{i,j}$ or $\Pi_{j,i}$ oracle. If $advantage^E(k)$ is non-negligible, then $Pr[E\ succeeds] = \frac{1}{2} + n(k)$, for some non-negligible function $n(k)$. At the same time,

$$Pr[E\ succeeds] = Pr[E\ succeeds|\Gamma_k]Pr[\Gamma_k] + Pr[E\ succeeds|\bar{\Gamma}_k]Pr[\bar{\Gamma}_k]. \quad (1)$$

Because H_2 is a random oracle and $\Pi_{i,j}^s$ remains unopened (by the definition of a fresh oracle), $Pr[E\ succeeds|\bar{\Gamma}_k] = 1/2$. Therefore,

$$\frac{1}{2} + n(k) \leq Pr[E\ succeeds|\Gamma_k]Pr[\Gamma_k] + \frac{1}{2}, \quad (2)$$

which implies that $Pr[E\ succeeds|\Gamma_k] \geq n(k)$, and $Pr[\Gamma_k] \geq n(k)$, are both non-negligible. Let $Pr[\Gamma_k] = n_1(k)$, for some non-negligible function $n_1(k)$. E can now be used to construct an adversary F that solves the CDH problem with non-negligible probability. Given P (the generator of the group $\mathbb{G}_1$), aP (akin to the X_1 value transmitted by i), and x_jP (j's public key), with randomly chosen $a \in_R \mathbb{Z}_q^*$ and $x_j \in_R \mathbb{Z}_q^*$, F must guess ax_jP. This is precisely an instance of the

CDH problem. The construction of F follows that in Case 1 of Theorem 9 in [7]. Let I be the set of entities with which E may communicate. F chooses a pair of entities $i, j \in_R I$, guessing that E, or an oracle other than $\Pi_{i,j}$ or $\Pi_{j,i}$ will query H_2 with $ax_jP\|ID_i\|ID_j\|T_1$ at some time T_1. F performs E's experiment and picks all entities' secret values at random, except of those of $\Pi_{i,j}$ and $\Pi_{j,i}$. F records the public keys of all entities and starts E. F answers all H_1 and H_2 queries at random, just like a random oracle. For all `Send`, `Reveal` and `Corrupt` queries made to oracles other than $\Pi_{i,j}$ or $\Pi_{j,i}$, F answers as an honest player would. If E asks i or j a `Corrupt` query then F gives up. When E asks $\Pi_{i,j}$ or $\Pi_{j,i}$ a `Send` query, instead of computing the value $sk = H_2(ax_jP\|ID_i\|ID_j\|T_1)$ (the key that would have been used by the oracle), F must select a random κ to represent sk, since F does not actually know ax_jP. Finally, if E sends a `Reveal` query to $\Pi_{i,j}$ or $\Pi_{j,i}$, then instead of revealing $H_2(ax_jP\|ID_i\|ID_j\|T_1)$, F responds with his guess, κ, at the session key.

Let $\tau_2(k)$ be a polynomial bound on the number of H_2 queries made by E and its oracles. F chooses $l \in \{1, \ldots, \tau_2(k)\}$, guessing that the l-th distinct input on which H_2 is queried will be $ax_jP\|ID_i\|ID_j\|T_l$, where T_l the time of the l-th query. We note that $T_1, T_2, \ldots, T_{\tau_2(k)}$ are known to the adversary because time-stamps are always transmitted in the clear, as are the entities' IDs. When the l-th distinct H_2 call is made (say, on input g), F stops and outputs g as its guess at $ax_jP\|ID_i\|ID_j\|T_l$ (or, equivalently, F outputs the first part of g corresponding to ax_jP, by discarding the identifiers and time stamp). If E halts before the l-th distinct input is queried then F gives up. Clearly, if the l-th distinct H_2 query made by E or its oracles is on an input that begins with ax_jP, then F wins at this experiment and has solved an instance of the CDH problem.

Of course, H_2 may have been queried on $ax_jP\|ID_i\|ID_j\|T_{l'}$ some time *before* the l-th distinct input, in which case F will have answered at random, and his answer may have been in contradiction to one of the keys that has been used by some $\Pi_{i,j}$ or $\Pi_{j,i}$ oracle. In such a case, E's behavior is not specified, thus E is not guaranteed to halt in that case. This problem can be circumvented as in [7], by letting $\tau_3(k)$ be a polynomial bound on E's runtime under ordinary circumstances and requiring that F gives up if he runs E for longer than $\tau_3(k)$, surmising that he must have missed an H_2 query with input $ax_jP\|ID_i\|ID_j\|T_{l'}$.

We conclude that the probability of F coming up with the correct value ax_jP is at least

$$\frac{n_1(k)}{\tau_1^2(k)\tau_2(k)}, \tag{3}$$

where $\tau_1(k)$ is a polynomial bound on the number of entities[10]. The quantity in (3) is non-negligible. We conclude that the polynomial time adversary F has succeeded in finding ax_jP given aP and x_jP, with non-negligible probability, contradicting the assumption that the CDH problem is hard.

[10] We note that the bound (3) can be improved further, because they adversary can limit its attention to H_2 queries made by E where the ID's included in the argument are ID_i and ID_j, as opposed to guessing over $\tau_1^2(k)$ queries.