A Study of Two-Party Certificateless Authenticated Key-Agreement Protocols

Colleen Swanson[1] and David Jao[2,⋆]

[1] David R. Cheriton School of Computer Science
[2] Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{c2swanso@cs,djao@math}.uwaterloo.ca

Abstract. We survey the set of all prior two-party certificateless key agreement protocols available in the literature at the time of this work. We find that all of the protocols exhibit vulnerabilities of varying severity, ranging from lack of resistance to leakage of ephemeral keys up to (in one case) a man-in-the-middle attack. Many of the protocols admit key-compromise impersonation attacks despite claiming security against such attacks. In order to describe our results rigorously, we introduce the first known formal security model for two-party authenticated certificateless key agreement protocols. Our model is based on the extended Canetti-Krawczyk model for traditional authenticated key exchange, except that we expand the range of allowable attacks to account for the increased flexibility of the attacker in the certificateless setting.

Keywords: key agreement, certificateless public key cryptography.

1 Introduction

Certificateless public key cryptography was introduced by Al-Riyami and Paterson [1] to serve as a middle ground between traditional public key cryptography based on PKI and the newer notion of identity-based cryptography in which a third party generates users' private keys. In certificateless public key cryptography, a user generates his private key by combining a secret value with a partial private key provided by the Key Generation Center (KGC). Similarly, to generate a public key, the user combines his secret value with public information from the KGC. Unlike in the case of identity-based cryptography, public keys are no longer easily computable by third parties, so they must be made available in some other way, such as via a public directory. However, once made available, the public keys do not need to be validated in any way; the security model for certificateless public key cryptography assumes that an adversary can replace public keys at will. Dent [4] has published a survey of the various certificateless public key encryption schemes that have been published since the introductory work of Al-Riyami and Paterson.

⋆ The authors were partially supported by NSERC.

In this work we focus on two-party key agreement schemes in the certificateless setting. In such schemes, two parties can establish a shared secret via a certificateless public key protocol. Work in the area of certificateless key agreement protocols is relatively limited. At the time this work was performed, we were aware of only five such schemes: one from the original Al-Riyami and Paterson paper [1], and four others by Mandt and Tan [11], Wang et al. [18], Shao [14], and Shi and Lee [15]. None of these works define a security model and therefore all of them lack a rigorous proof of security.

The contributions of this article are twofold. In the first part of the paper, we introduce a formal security model for certificateless authenticated key exchange protocols. Our security model is an extended version of the extended Canetti and Krawczyk model [7] for traditional authenticated key exchange. The modifications consist of enhancing the powers of the adversary to take into account the greater capabilities afforded in the certificateless setting. In the second part of the paper, we examine all five extant two-party certificateless authenticated key agreement protocols in the context of our security model. We find that all existing certificateless protocols allow for practical attacks of varying severity, ranging from relatively minor attacks involving leaked ephemeral secrets, up to (in one case) an outright man-in-the-middle attack. These results indicate that more work is required in order to fully realize the benefits of the certificateless paradigm in the context of key agreement.

We remark that, based upon an earlier unpublished version of this work [16], Lippold et al. [9] have published a subsequent security model for certificateless key agreement schemes, as well as a two-party certificateless key agreement scheme which is provably secure in their model (as well as ours). In Section 4 we explain the relationship between our security model and theirs. The provable security of their protocol means that, in a theoretical sense, the question of constructing a provably secure certificateless key agreement scheme in the context of a suitable security model has been settled. However, their protocol is slow, requiring each party to perform 10 pairing operations in order to compute the common key (or 5 at the cost of a stronger assumption). Therefore, it remains an open question whether there exists a provably secure certificateless key agreement protocol having minimal performance penalty compared to the alternatives. We discuss these and other issues in Section 6.

2 Background

We recall the standard terminology of key agreement protocols. A *key establishment protocol* is a protocol in which two or more parties gain access to a shared secret. If the shared secret is a function of information provided by and/or associated with each party (as opposed to the case where only one party is involved in choosing and/or obtaining the secret), we say the protocol is a *key agreement* protocol (KAP). We concern ourselves mainly with the two-party *dynamic* key agreement setting, wherein the established key varies with each execution. We refer to a protocol run as a *session*, and each message transmitted from one

party to another as a *flow*. The shared secret resulting from a session is generally called (or used to determine) a *session key*. Protocols generally assume users (or pairs of users) have *long-term keys*, which are static secrets that are usually precomputed and stored securely. These are often used in conjunction with randomized secret input, which we refer to as *ephemeral* or *short-term* keys. Many key agreement protocols also assume the existence of a centralized server, known as a *Key Generation Center (KGC)*. We refer to this entity's secret information as the *master secret key*. We assume that the KGC communicates with users via a secure channel, whereas protocol flows are sent via an open channel. That is, eavesdroppers have no access to KGC/user communication, but can easily read anything sent between protocol participants.

2.1 Bilinear pairings

All of the protocols that we discuss make use of bilinear pairings. Let q be a prime, G a cyclic additive group of order q generated by P, and G_T a multiplicative group of order q. Let $e\colon G \times G \to G_T$ be an admissible pairing, namely, one where e satisfies the following properties:

1. *Bilinearity*: $\forall P, Q, R \in G$ we have both $e(P+Q,R) = e(P,R)e(Q,R)$ and $e(P,Q+R) = e(P,Q)e(P,R)$.
2. *Non-degeneracy*: For all $P \neq 1_G$, we have $e(P,P) \neq 1_{G_T}$.
3. The pairing is efficiently computable.

The hardness assumption required for the pairing varies depending on the protocol. The survey article of Boyen [3] contains a list of the standard assumptions used in pairing-based protocols.

3 Security Attributes

Authenticated key agreement protocols satisfy the property that an entity is only able to compute a shared secret key if it holds the claimed identity. In particular, key agreement protocols should not allow an adversary to impersonate a user without that user's private key. The following security attributes apply to key agreement protocols in general:

- *Known session key security*: Key agreement protocols should be dynamic: each protocol run should result in a unique session key. An attacker who learns a given number of session keys should not be able to discover other session keys.
- *Forward secrecy*: Given the long-term private keys of one or more users, it is clearly desirable that an attacker not be able to determine previously established session keys. *Perfect forward secrecy* implies an attacker, even armed with all participants' long-term private keys, cannot determine old session keys. *Partial forward secrecy* implies an attacker armed with some, but not all, participants' long-term private keys cannot determine old session keys.

Similarly, *KGC forward secrecy* deals with the case in which the attacker has the master secret key. *Weak perfect forward secrecy* deals with the case where all long-term private keys are known, but the attacker was not actively involved in choosing ephemeral keys during the sessions of interest. It has been shown by Krawczyk [6] that no 2-flow authenticated key agreement protocol can do better than this weaker version of forward secrecy.

- *Unknown key-share security*: It should be impossible to coerce A into thinking he is sharing a key with B, when he is actually sharing a key with another (honest) user C. That is, it should not be possible for A to believe he is sharing a key with $B \neq C$, while C correctly thinks the key is shared with A.
- *Resilience to key-compromise impersonation (KCI)*: If the long-term private key of user A is compromised, the attacker should not be able to impersonate another user B to A. Obviously, if a long-term private key of A is compromised, we wish to replace this key as soon as possible, as the attacker can certainly impersonate A to any other user; this property is nevertheless important in the sense that it minimizes the damage until the user can detect that his key has been compromised.
- *Resistance to leakage of ephemeral keys*: If the attacker has access to the ephemeral keys of a given protocol run, he should be unable to determine the corresponding session key. As argued by Menezes and Ustaoglu [12], adversaries may gain access to this information through a side-channel attack or use of a weak random number generator; alternatively this information might be stored insecurely. We refer to such attacks as *known ephemeral key attacks*.

Additional security requirements arise in the case of certificateless key agreement protocols. Since public keys are not validated as in ID-based schemes or traditional PKI, we must assume that an adversary can replace public keys at will, and this attack must be incorporated into the security model. Now, if a KGC replaces public keys, it will be able to impersonate any user, since it can easily compute the corresponding private key. Thus, for all certificateless schemes, the KGC can launch a man-in-the-middle attack. For this reason, security models for certificateless schemes generally assume that the KGC never replaces public keys. Al-Riyami and Paterson [2] argue that this amounts to roughly the same amount of trust that is invested in a certificate authority (CA) in a traditional PKI. They make the point that, while often not stated explicitly, we usually trust that CAs do not produce certificates binding arbitrary public keys to a given identity. In any case, CL-PKC amounts to less trust than in an ID-based scheme, where the KGC has access to users' private keys by definition.

One way to avoid the issue of the KGC replacing public keys is to bind a user's public and private keys, as noted in Al-Riyami and Paterson [1]. This technique requires the user to send his fixed public key to the KGC, which is then incorporated into the partial private key. The result is that there can be only one working public key per user, so the existence of more than one implies that the KGC created more than one partial private key binding the user to

different public keys. In fact, with this binding technique, the partial private keys do not need to be kept secret. The corresponding unique public key was computed with the user's secret value—this value is necessary to compute the full private key and cannot be determined from the exposed partial private key. While this binding technique has certain advantages, it comes at the added cost of reduced flexibility. With the binding technique in place, users must establish their public key before receiving their partial private key from the KGC.

4 Security Model

In this section, we give an extension of the extended Canetti-Krawczyk (eCK) [7] model suitable for the certificateless key agreement setting. In particular, the eCK model (and our model as well) captures all of the basic security attributes mentioned in Section 3.

We note that Al-Riyami and Paterson give a security definition relevant to certificateless encryption in [1, 2], but our key agreement model is based on the eCK model and is not the natural extension of their definition to key establishment protocols. The treatment of the long-term secret information of the user is different in Al-Riyami and Paterson's definition, as their adversary is not allowed to know only part of a user's private key, an issue which restricts the treatment of leakage of ephemeral information. In particular, Al-Riyami and Paterson's model does not account for the attacks given in the following section, so this model is not sufficient when we consider several real-world attacks. Nevertheless there are some similarities, and we owe the general adversarial model to Al-Riyami and Paterson: we consider two possible types of adversaries, namely those without the master secret key, who can replace public keys at will, and those with the master secret key, who are not allowed to replace public keys at any time.

Informally, we refer to these types of attackers as either *outside* or *inside* attackers, in the following sense:

- An adversary is an *outside attacker* if the adversary does not have the master secret key; an outside attacker is able to replace public keys of users.
- An adversary is an *inside attacker* if the adversary has access to the master secret key; an inside attacker cannot replace public keys of users.

The formal adversarial model is as follows.

As in the eCK model, we consider a finite set of parties $P_1, P_2, \ldots, P_n$ modeled by probabilistic Turing machines. The adversary, also modeled by a probabilistic Turing machine, controls all communication—parties give outgoing messages to the adversary, who has control over their delivery via the Send query. Parties are activated by Send queries, so the adversary has control over the creation of protocol sessions, which take place within each party. We call the initiator of a session the *owner*, the responder the *peer*, and say both are *partners* of the given session. We define a *conversation* for a given session to be the ordered concatenation of all messages (both incoming and outgoing), and say two sessions s and s' have *matching conversations* if the outgoing messages of one are the

incoming messages of the other, and vice versa. In particular, we assume that the public keys of the parties are a part of the message flows.[3]

We borrow the queries EphemeralKeyReveal and SessionKeyReveal from the eCK model. The EphemeralKeyReveal($\mathfrak{s}$) query allows the adversary to obtain the ephemeral private key of the session $\mathfrak{s}$; this is not equivalent to issuing the query EphemeralKeyReveal on the session matching to $\mathfrak{s}$ (if it exists), as only the ephemeral information chosen by the session owner is revealed. The SessionKeyReveal($\mathfrak{s}$) query allows the adversary to obtain the session key for the specified session $\mathfrak{s}$ (so long as $\mathfrak{s}$ holds a session key).

In addition to these queries, we allow the adversary the queries RevealMasterKey, ReplacePublicKey, RevealPartialPrivateKey(party), and RevealSecretValue(party). The adversary can gain access to the master secret key via the query RevealMasterKey and replace the public key of a given party via ReplacePublicKey query. Unlike in the eCK model, this does not mean the adversary has control over the party. Instead, it implies that all other parties will use the adversary's version of the party's public key, while the given party will continue to use the correct public key in any calculations.[4] The RevealPartialPrivateKey(party) query gives the adversary access to the given party's partial private key, which is generated from the master secret key. (Note that this command is redundant if the RevealMasterKey query has been issued.) Lastly, the RevealSecretValue(party) query gives the adversary access to the party's chosen secret value (which is used to generate the party's public key). We assume that an adversary cannot issue a RevealSecretValue query against a party which has already received the ReplacePublicKey query.

We consider an adversary-controlled party to be one against which the adversary has issued both the ReplacePublicKey and RevealPartialPrivateKey queries. If the RevealMasterKey query has been issued, any party issued the ReplacePublicKey query is considered to be adversary-controlled; in this way, we capture the intent of the requirement that adversaries holding the master key should not be allowed to replace public keys. In particular, we say a party that is not adversary-controlled is *honest*. Formally, we define a *fresh* session as follows:

Definition 1. Let $\mathfrak{s}$ be a completed session owned by party P_i with peer P_j, both of whom are honest. Let $\mathfrak{s}^*$ denote the matching session (if such a session exists). We say $\mathfrak{s}$ is *fresh* if none of the following conditions hold, where E denotes the adversary:

1. E issues a SessionKeyReveal($\mathfrak{s}$) or SessionKeyReveal($\mathfrak{s}^*$) query (provided $\mathfrak{s}^*$ exists);

[3] The thesis version of this work [16], as pointed out by Lippold et al. [9], did not include public keys in the definition of a matching conversation. This is an oversight in our original model, as it does not allow an adversary to replay conversations with replaced public keys without detection.

[4] In Lippold et al. [9]'s model, the given party will use the replaced public key instead of his chosen key. While their model is strictly stronger than ours because of this, we feel it is a more natural choice to assume that a party knows his own public key.

2. $\mathfrak{s}^*$ exists and E either makes queries:
   (a) both $\mathsf{RevealPartialPrivateKey}(P_i)$ and $\mathsf{RevealSecretValue}(P_i)$ as well as $\mathsf{EphemeralKeyReveal}(\mathfrak{s})$ or
   (b) both $\mathsf{RevealPartialPrivateKey}(P_j)$ and $\mathsf{RevealSecretValue}(P_j)$ as well as $\mathsf{EphemeralKeyReveal}(\mathfrak{s}^*)$;
3. No matching session $\mathfrak{s}^*$ exists and E either makes queries:
   (a) both $\mathsf{RevealPartialPrivateKey}(P_i)$ and $\mathsf{RevealSecretValue}(P_i)$ as well as $\mathsf{EphemeralKeyReveal}(\mathfrak{s})$ or
   (b) both $\mathsf{RevealPartialPrivateKey}(P_j)$ and $\mathsf{RevealSecretValue}(P_j)$.

This definition encompasses both types of adversaries. In the case where the adversary has issued the $\mathsf{RevealMasterKey}$ query, he cannot issue replace public key queries without making the involved parties dishonest. Moreover, as this adversary automatically has access to users' partial private keys, he is assumed unable to issue both the $\mathsf{RevealSecretValue}(P_i)$ and $\mathsf{EphemeralKeyReveal}(\mathfrak{s})$ or the $\mathsf{RevealSecretValue}(P_j)$ and $\mathsf{EphemeralKeyReveal}(\mathfrak{s}^*)$ queries (provided that $\mathfrak{s}^*$ exists).

As in the eCK model, we allow the adversary E a single $\mathsf{Test}(\mathfrak{s})$ query, which can be issued at any stage to a completed, fresh session $\mathfrak{s}$. A bit b is then picked at random. If $b = 0$, the test oracle reveals the session key, and if $b = 1$, it generates a random value in the key space. E can continue to issue queries as desired, with the requirement that the test session remain fresh. At any point, the adversary can try to guess b. Let $\text{GoodGuess}^E(k)$ be the event that E correctly guesses b, and $\text{Advantage}^E(k) = \max\left\{0, \left|\Pr[\text{GoodGuess}^E(k)] - \frac{1}{2}\right|\right\}$, where k is a security parameter. We are now ready to formally define our notion of a secure session.

Definition 2. We say a certificateless key establishment protocol is *secure* if the following conditions hold:

1. If honest parties have matching sessions and no $\mathsf{ReplacePublicKey}$ queries have been issued, these sessions output the same session key (except with negligible probability).
2. For any polynomial time adversary E, $\text{Advantage}^E(k)$ is negligible.

5 Attacks

In this section we explore several two-party certificateless authenticated key agreement protocols from the literature, as well as one "self-certified" protocol (which on closer analysis actually appears to be certificateless). We first establish notation and give a summary of the protocols in Section 5.1. We then discuss each protocol in detail and present relevant attacks, modeled within the framework given in Section 4. We pay particular attention to the existence of key compromise impersonation attacks and whether or not the protocols have resistance to leakage of ephemeral keys, and in one case we show the protocol to be entirely insecure.

Table 1. Parameters for user i

Scheme	P_i	Q_i	S_i	X_i
AP [1]	$\langle x_i P, x_i P_{\mathrm{KGC}} \rangle$	$h(\mathrm{ID}_i)$	sQ_i	$x_i S_i$
MT [10, 11]	$x_i P$	$h(\mathrm{ID}_i)$	sQ_i	$(s + x_i)Q_i$
WCW [18]	$x_i P$	$h(\mathrm{ID}_i)$	sQ_i	$\langle x_i, S_i \rangle$
Shao [14]	$x_i P$	$h'(\mathrm{ID}_i, P_i)$	sQ_i	$\langle x_i, S_i \rangle$
SL[15]	$e(P, x_i P)$	$H(\mathrm{ID}_i)$	$\frac{1}{Q_i + s} P$	$x_i S_i$

5.1 Protocol Summaries

In all of the following protocols, the Key Generation Center (KGC) has master secret key $s \in \mathbb{Z}_q^*$ and master public key $P_{\mathrm{KGC}} = sP$. The system parameters $\langle q, G, G_T, e, P, P_{\mathrm{KGC}} \rangle$ are as in Section 2.1. Let $k \in \mathbb{N}$ denote the number of bits in the shared session key. We will need the following hash functions and key derivation functions:

$H\colon \{0,1\}^* \to \mathbb{Z}_q^*$, $H'\colon G_T \to \mathbb{Z}_q^*$, $h\colon \{0,1\}^* \to G$, $h'\colon \{0,1\}^* \times G \to G$, and $h''\colon G_T \times G \to \{0,1\}^k$, $\mathrm{kdf}_{\mathrm{MT}}\colon G_T \times G \times G \to \{0,1\}^k$, $\mathrm{kdf}_{\mathrm{WCW}}\colon \{0,1\}^* \times \{0,1\}^* \times G_T \times G \times G \to \{0,1\}^*$, $\mathrm{kdf}_{\mathrm{Shao}}\colon G_T \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^k$, and $\mathrm{kdf}_{\mathrm{SL}}\colon G_T \to \{0,1\}^k$.

The certificateless protocols we study are given in Tables 1, 2, and 3, and include all certificateless protocols published at the time of our work, namely those of Al-Riyami and Paterson (AP) [1], Mandt and Tan (MT) [10, 11], Wang, Cao, and Wang (WCW) [18], Shao [14], and Shi and Li (SL) [15].

In Table 1, we give a summary of user parameters for each of the protocols. For a given user i, we use P_i to denote the public key of i and Q_i to denote the mapping of ID_i into G. Each user has two pieces of secret information, a user-chosen secret value $x_i \in_R \mathbb{Z}_q^*$ and a partial private key $S_i \in G$, provided by the KGC. We use X_i to denote user i's full private key, which is formed from a combination of the user's secret information S_i and x_i.

We give the shared secret and session key computation information in Tables 2 and 3, respectively. For each protocol we have users A and B, with communication consisting of one message flow each. User A picks $a \in_R \mathbb{Z}_q$ and sends T_A to B, and B similarly chooses $b \in_R \mathbb{Z}_q$ and sends T_B to A. Some of the protocols specifically send user public keys as part of the message flows, and some assume the existence of a public directory for this purpose. For ease of representation, we do not specify which. Similarly, we do not include all of the computational checks that users perform during the protocol, but note that each protocol has group membership checks in place to thwart subgroup membership attacks, such as those of Lim and Lee [8], and the AP protocol includes a check to ensure that user public keys are of the correct form.

5.2 Al-Riyami and Paterson (AP) protocol

As pointed out by Mandt [10], the AP protocol does not have resistance to leakage of ephemeral keys. However, we can easily fix this by computing the

Table 2. Shared secret computation

Scheme	T_i		Shared Secret K
AP [1]	T_A	aP	$e(X_B, T_A)e(X_A, T_B)$
	T_B	bP	$e(Q_B, x_A P_{\mathrm{KGC}})^a e(X_A, T_B)$
MT [11]	T_A	aP	$e(Q_B, P_{\mathrm{KGC}} + P_B)^a e(Q_A, P_{\mathrm{KGC}} + P_A)^b$
	T_B	bP	$e(Q_B, P_{\mathrm{KGC}} + P_B)^a e(X_A, T_B)$
WCW [18]	T_A	aP	$e(Q_A, Q_B)^s$
	T_B	bP	$e(S_A, Q_B)$
Shao [14]	T_A	aP_B	$H'(e(Q_A, Q_B)^s)abP$
	T_B	bP_A	$H'(e(S_A, Q_B))ax_A^{-1} \pmod q T_B$
SL [15]	T_A	$a(Q_B + P_{\mathrm{KGC}})$	$P_A^b P_B^a$
	T_B	$b(Q_A + P_{\mathrm{KGC}})$	$e(T_B, X_A)P_B^a$

Table 3. Session key computation. Here K denotes the shared secret from Table 2.

Scheme	Session Key
AP [1]	$h''(K\|\|abP)$
MT [11]	$\mathrm{kdf}(K\|\|abP\|\|x_A x_B P)$
WCW [18]	$\mathrm{kdf}(\mathrm{ID}_A, \mathrm{ID}_B, K, ax_B P, bx_A P)$
Shao [14]	$\mathrm{kdf}(K\|\|\,\mathrm{ID}_A\,\|\|\,\mathrm{ID}_B)$
SL [15]	$\mathrm{kdf}(K)$

session key to be $h''(K||abP||x_A x_B P)$ instead of $h''(K||abP)$, modifying h'' as necessary (where Alice computes $x_A(x_B P)$ and Bob computes $x_B(x_A P)$).

Our fix is not quite as strong as we would like, however. In the context of our formal model from Section 4, an outside adversary (one who has not issued the RevealMasterKey query) can still mount an attack on A. He simply issues the ReplacePublicKey query on B and uses the EphemeralKeyReveal query on both the test session and its matching session. The initial vulnerability of the protocol to the leakage of ephemeral keys allows the attacker to compute K. He can compute A's version of $x_A x_B P$ by using his knowledge of the replaced public key. Specifically, suppose he chooses to replace B's public key with $\langle x'_B P, x'_B sP\rangle$ for some $x'_B \in_R \mathbb{Z}_q^*$. Then A will compute $x_A x_B P$ as $x_A(x'_B P) = x'_B(x_A P)$. It is clear that the adversary will be able to distinguish the session key held by A from a randomly chosen element of the keyspace.

5.3 Mandt and Tan (MT) protocol

The MT protocol [10, 11] was designed to satisfy all the security properties of the Al-Riyami and Paterson protocol, as well as the additional property of resistance to leakage of ephemeral keys. Mandt and Tan argue heuristically that the MT protocol has this property, as well as known session key security, weak forward secrecy, and resistance to key compromise impersonation, unknown key share, and key control attacks. We now show that resistance to leakage of ephemeral information does not hold, and that the protocol actually admits both a key

compromise impersonation and a known ephemeral key attack. The KCI attack mentioned below was also independently discovered by Xia et al. [19].

Mandt and Tan provide two variations of the basic protocol given above, one in which the protocol participants use separate KGCs and one which provides key confirmation. The former is almost identical to the basic protocol, with the substitution of the different master keys where appropriate. The latter uses a MAC keyed under a value related to that of the session key, i.e., derived using a different key derivation function on the same inputs. Both versions are vulnerable to the attacks outlined below. We have included only the basic version of the protocol as described in [10] to improve readability.

We first show the protocol is vulnerable to a key compromise impersonation attack from an outside attacker. In fact, it suffices for the adversary Eve (E) to know a user's secret value x_i; she does not need the partial private key provided by the KGC. As the flows in the protocol are symmetric, it does not matter whether the adversary attempts to impersonate the initiator or responder of the protocol run. Formally, we describe this attack sequence on a session held by Alice as $\mathsf{RevealSecretValue}(A)$ and $\mathsf{ReplacePublicKey}(B)$ where no matching session exists, i.e., the adversary uses the Send query to send messages purportedly from B to A.

Assume Eve has access to x_A. She impersonates Bob to Alice by selecting $\beta, b \in \mathbb{Z}_q^*$ and sending $P_B^* = -P_{\mathrm{KGC}} + \beta P$ for B's public key and $T_B = bP$ as usual. Since $P_B^* \in G^*$, Alice computes

$$\begin{aligned} K &= e(Q_B, P_{\mathrm{KGC}} + P_B^*)^a e(X_A, T_B) \\ &= e(Q_B, P_{\mathrm{KGC}} - P_{\mathrm{KGC}} + \beta P)^a e((s + x_A)Q_A, bP) \\ &= e(Q_B, \beta P)^a e(bQ_A, (s + x_A)P) \\ &= e(\beta Q_B, aP) e(bQ_A, P_{\mathrm{KGC}} + P_A). \end{aligned}$$

We denote by K_A the value of K computed by Alice above; that is, Alice thinks K_A is the correct value of K. Alice then derives the session key from $\mathrm{kdf}(K_A || aT_B || x_A P_B)$.

As Eve chooses both β and b, she can compute K_A and $aT_B = bT_A$. Note that we have not needed knowledge of x_A up to this point. The only reason we need x_A is to compute $x_A P_B$ to input into the kdf, so this term is the only preventative measure against a man-in-the-middle attack similar to the KCI attack above. We see no clever substitution for P_B which allows for both the calculation of $e(Q_B, P_{\mathrm{KGC}} + P_B)^a$ and $x_A P_B$, however. The heuristic argument against KCI attacks given in [10] fails to consider the scenario where the public key for B is replaced. The subsequent paper by Mandt and Tan [11] recognizes the possibility of replacing public keys, but still fails to identify the above attack.

Mandt and Tan also claim the protocol has the property of resistance to leakage of ephemeral keys. However, it is an easy matter to mount an outsider man-in-the-middle attack if given a and b. Suppose Eve substitutes $P_A^* = \alpha P$ for P_A and $P_B^* = \beta P$ for P_B in the protocol run, where $\alpha, \beta \in_R \mathbb{Z}_q^*$. From above we

have that A and B will compute $K_A = e(Q_B, P_{\text{KGC}} + P_B^*)^a e(Q_A, P_{\text{KGC}} + P_A)^b$ and $K_B = e(Q_B, P_{\text{KGC}} + P_B)^a e(Q_A, P_{\text{KGC}} + P_A^*)^b$, respectively, so Eve will have no problem calculating K_A and K_B if she knows a and b.

Moreover, we have $x_A P_B^* = x_A \beta P = \beta P_A$, so Eve will establish the session key $\text{kdf}(K_A || aT_B || x_A P_B^*) = \text{kdf}(K_A || abP || \beta P_A)$ with A. Similarly she will establish $\text{kdf}(K_B || abP || \alpha P_B)$ as the session key with B. Thus the protocol fails to have resistance to leakage of ephemeral keys against outside attackers. If, on the other hand, the attacker is passive or cannot replace public keys, the protocol remains secure. The variants of the protocol are similarly vulnerable. Note that this is essentially the same attack mentioned in Section 5.2 on the improved version of the Al-Riyami Paterson KAP.

Interestingly, the MT protocol is almost identical to the protocol of Wang and Cao [17]. The main difference between the two protocols is that in the latter, the private keys are bound to the public keys, so the attacks presented above are not possible in an ideal protocol specification of Wang and Cao's protocol, whereas the MT protocol allows adversaries to easily replace public keys. We achieve the same scheme if we apply the binding technique of Al-Riyami and Paterson [1] to Mandt's protocol, although for cheating to be evident, we must require the partial private keys (or users' certificates) to be public.

5.4 Wang et al. (WCW) protocol

Wang et al. [18] claim the WCW protocol is secure against both man-in-the-middle attacks mounted by the KGC and KCI attacks. Since all certificateless key agreement protocols are vulnerable to a KGC man-in-the-middle attack (if the KGC can replace public keys), the first claim is certainly false. We show the second claim to be false by presenting a KCI-attack below; this attack was also independently discovered by Meng and Futai [13]. We mention that Wang et al. also claim their protocol has known-key security, weak partial and KGC forward secrecy, unknown key-share resistance, and key control.

We observe that use of the static shared secret K prevents the formal attack outlined in Section 5.2, where knowledge of the matching sessions' ephemeral keys and one public key replacement allows a successful attack. This static shared secret implies a successful attacker must have access to at least one of the participating party's partial private keys. However, the protocol does not guard against an adversary who, for a test session $\mathfrak{s}$ with owner A and matching session $\mathfrak{s}^*$, issues the queries $\mathsf{RevealPartialPrivateKey}(A)$, $\mathsf{EphemeralKeyReveal}(\mathfrak{s})$, and $\mathsf{EphemeralKeyReveal}(\mathfrak{s}^*)$. The adversary will be able to compute the session key $\text{kdf}(\text{ID}_A, \text{ID}_B, K, ax_B P, bx_A P)$.

We mount an outsider KCI attack as follows. As with the KCI attack on the Mandt protocol, we can express this attack using the formal terminology of Section 4. The adversary chooses a test session owned by B and takes advantage of the queries Send, $\mathsf{RevealPartialPrivateKey}(B)$ and $\mathsf{ReplacePublicKey}(A)$. The (informal) details follow.

We assume that Eve is attempting to impersonate Alice to Bob, where Alice is the initiator and Bob is the responder. For such an attack, we would generally

assume our attacker Eve has access to all of B's private key, that is, both x_B and S_B. Here we show that it is sufficient for Eve to have S_B.

Eve proceeds by sending $T_A = aP$ as usual (for her own choice of $a \in_R \mathbb{Z}_q^*$). She completes the attack by replacing A's public key entry with $P_A^* = \alpha P$ for some $\alpha \in_R \mathbb{Z}_q^*$.

Note that Eve can easily compute $K = e(S_B, Q_A)$, as she has access to S_B. Recall that B will use $x_B T_A$ and bP_A^* in his computation of the secret key. Since $x_B T_A = aP_B$ and $bP_A^* = b\alpha P = \alpha T_B$, Eve can compute these as well, because a and α were chosen by her, the term P_B is public, and B sends T_B to A during the protocol run. Thus Eve can compute $K_B = \mathrm{kdf}(\mathrm{ID}_A, \mathrm{ID}_B, K, x_B T_A, bP_A^*)$, as desired. It is worth stressing that Eve cannot succeed without knowledge of S_B, as without it she cannot compute K.

5.5 Shao protocol

The Shao [14] protocol purportedly provides weak forward secrecy, known-key security, and resilience to a "masquerade attack," which refers to KCI attacks where not *all* of the user's private key is compromised. We show that the latter claim is false by providing a KCI attack that only requires the adversary to have access to part of the user's private key. Although Shao claims his protocol is self-certified, the scheme is much closer to a certificateless protocol than Girault's [5] notion of a self-certified key agreement protocol. In contrast to Girault's model, the KGC has partial key escrow (in that it computes the partial private keys) and requires a secure channel between itself and the users. Also, users cannot be sure that they have the correct public key of another party, so the adversarial model is equivalent to that of certificateless schemes.

Moreover, although at first glance it seems that Shao's protocol has applied the binding technique of Al-Riyami and Paterson mentioned in Section 2, given his definition of Q_i for user i, this is not quite the case. The protocol relies in an essential way on the secrecy of the partial private keys; if we make these keys public, the scheme reduces to the basic Diffie-Hellman KAP (with a single extra term $H'(e(S_A, Q_B))$ that anyone can compute).

We observe that this protocol, like that of Section 5.4, does not hold up to the following formal attack. Letting $\mathfrak{s}$ denote the test session with owner A and matching session $\mathfrak{s}^*$, suppose the adversary issues the $\mathsf{RevealPartialPrivateKey}(A)$, $\mathsf{EphemeralKeyReveal}(\mathfrak{s})$, and $\mathsf{EphemeralKeyReveal}(\mathfrak{s}^*)$ queries. The adversary will be able to compute $H'(e(S_A, Q_B))abP$, and hence the session key as well.

Let us now consider Shao's claim that the protocol is secure provided not all of the user's private key (x_i, S_i) is compromised. We show the protocol is in fact vulnerable in the scenario where S_i is known, but x_i and s remain secure.

We launch an outsider key compromise impersonation attack on Alice with the knowledge of S_A, but not x_A, as follows. Since knowing $S_A = sQ_A = sH'(\mathrm{ID}_A, x_A P)$ is not the same as knowing s, replacing public keys is permissible in this scenario. As the protocol messages are symmetric, there is a corresponding attack on Bob, and thus it does not matter whether or not the attacker

initiates the protocol run. The formal queries needed for this attack are similar to the KCI attack outlined in Section 5.4, so we do not mention them here.

Our attacker Eve replaces Bob's public key with $P_B^* = \beta P$ for $\beta \in_R \mathbb{Z}_q^*$ of her choosing. She then follows the protocol and sends ID_B and $T_B = bP_A$ for some $b \in_R \mathbb{Z}_q^*$. Alice will then compute Q_B as $Q_B = h'(\mathrm{ID}_B, P_B^*)$, and $H'(e(S_A, Q_B))ax_A^{-1} \pmod q$.

Alice calculates the session secret as

$$\begin{aligned} K &= H'(e(S_A, Q_B))ax_A^{-1}bP_A \\ &= H'(e(S_A, Q_B))bax_A^{-1}x_AP \\ &= H'(e(S_A, Q_B))baP. \end{aligned}$$

We see that Eve can compute $H'(e(S_A, Q_B))b$, as she possesses S_A and chooses b herself. Moreover, since A sends $T_A = a\beta P$ in the first round (and Eve knows β), Eve can compute aP and thus K.

5.6 Shi Li (SL) protocol

In the SL protocol [15], the session key is derived directly from $P_A^b P_B^a$, so this protocol certainly does not have resistance to leakage of ephemeral keys. The authors claim that this protocol provides implicit key authentication, known session key security, weak partial forward secrecy, key compromise impersonation resistance, and unknown key share resistance. We show the protocol fails to provide implicit key authentication by demonstrating a man-in-the-middle attack by an outside attacker.

Our attacker Eve intercepts Alice's $\langle T_A, P_A \rangle$ and instead sends $\langle T_A^*, P_A^* \rangle$ to Bob. Here $T_A^* = a^*(H(\mathrm{ID}_B)P + P_{\mathrm{KGC}})$ and $P_A^* = e(\alpha(H(\mathrm{ID}_A)P + P_{\mathrm{KGC}}), P)$ for $a^*, \alpha \in_R \mathbb{Z}_q^*$ of Eve's choosing.

Similarly Eve replaces Bob's message $\langle T_B, P_B \rangle$ with $\langle T_B^*, P_B^* \rangle$, where $T_B^* = b^*(H(\mathrm{ID}_A)P + P_{\mathrm{KGC}})$ and $P_B^* = e(\beta(H(\mathrm{ID}_A)P + P_{\mathrm{KGC}}), P)$ for $b^*, \beta \in_R \mathbb{Z}_q^*$ of her choosing.

Notice that $P_A^* \in G_T$, so Bob will compute

$$\begin{aligned} K_B &= e(T_A^*, X_B)(P_A^*)^b \\ &= e\left(a^*(H(\mathrm{ID}_B)P + P_{\mathrm{KGC}}), \frac{x_B}{H(\mathrm{ID}_B) + s}P\right) e(\alpha(H(\mathrm{ID}_A)P + P_{\mathrm{KGC}}), P)^b \\ &= e(a^*P, x_BP)e(b(H(\mathrm{ID}_A)P + P_{\mathrm{KGC}}), \alpha P) \\ &= P_B^{a^*} e(T_B, \alpha P). \end{aligned}$$

As Eve chooses both a^* and α, she can compute K_B. Similarly, Eve will be able to compute Alice's key $K_A = P_A^{b^*} e(T_A, \beta P)$. We have therefore shown the protocol to be insecure. The corresponding formal attack on the protocol is modeled by picking a test session with owner A and using the $\mathsf{ReplacePublicKey}(B)$ and Send queries to alter the messages sent by B to A. As shown above, the adversary will be able to compute the session key. Interestingly, this attack fails if

we transform this protocol into a certificate-based protocol, whereby the public keys of users are bound to the corresponding partial private keys and the latter are used as public certificates.

6 Conclusion

Our work demonstrates that all existing CL-KAPs are insecure to some extent in the sense of the security model of Section 4. In our opinion, this model is a natural one, and our findings provide motivation for developing new CL-KAPs meeting this security definition. We remark that, since the initial version of this work [16], new protocols have appeared [13, 9] which were designed to address some of the shortcomings of the earlier protocols.

The existence of practical CL-KAPs remains an interesting open question, given that these schemes are designed to avoid both the key escrow problem and the high management costs of certificate distribution, storage, verification, and revocation present in public key infrastructures. These protocols also have the added advantage of flexibility—the user can generate his public key before *or* after receiving his partial private key. Consequently, while applying Al-Riyami and Paterson's binding technique fixes some of the security vulnerabilities mentioned in Section 5, doing so limits the advantages gained by using a certificateless scheme in the first place.

Although Lippold et al. [9] have settled the question of whether a CL-KAP exists that is secure in the extended eCK model given in Section 4 (as well as their strengthened version of this model), the question of whether a computationally practical scheme exists remains. Lippold et al.'s scheme, which is secure given the bilinear and computational Diffie-Hellman assumptions, requires 10 pairing computations per party; even the version relying on the gap bilinear Diffie-Hellman assumption is expensive, requiring 5 pairing computations per party. In addition, the security of their scheme is proven using the random oracle model, so it remains an open question to devise a scheme secure in the standard model.

References

1. Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In C. S. Laih, editor, *Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer Berlin / Heidelberg, 2003.
2. Sattam S. Al-Riyami and Kenneth G. Paterson. CBE from CLE-PKE: A generic construction and efficient schemes. In *PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 398–415. Springer Berlin / Heidelberg, 2005.
3. Xavier Boyen. The uber-assumption family – a unified complexity framework for bilinear groups. In *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Berlin: Springer-Verlag, 2008.
4. Alexander W. Dent. A survey of certificateless encryption schemes and security models. *Int. J. Inf. Secur.*, 7(5):349–377, 2008.
5. Marc Girault. Self-certified public keys. In *Eurocrypt '91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer Berlin / Heidelberg, 1991.

6. Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *Crypto 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer Berlin / Heidelberg, 2005.
7. Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 2007.
8. Chae Hoon Lim and Pil Joon Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263. Springer Berlin / Heidelberg, 1997.
9. Georg Lippold, Colin Boyd, and Juan González Nieto. Strongly secure certificateless key agreement. In *Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 206–230. Berlin: Springer-Verlag, 2009.
10. Tarjei K. Mandt. Certificateless authenticated two-party key agreement protocols. Master's thesis, Gjøvik University College, Department of Computer Science and Media Technology, 2006.
11. Tarjei K. Mandt and Chik How Tan. Certificateless authenticated two-party key agreement protocols. In *Advances in Computer Science - ASIAN 2006. Secure Software and Related Issues*, volume 4435 of *Lecture Notes in Computer Science*, pages 37–44. Springer Berlin / Heidelberg, 2008.
12. Alfred Menezes and Berkant Ustaoglu. Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard. In *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 261–270, New York, NY, USA, 2008. ACM.
13. Gao Meng and Zhang Futai. Key-compromise impersonation attacks on some certificateless key agreement protocols and two improved protocols. *Education Technology and Computer Science, International Workshop on*, 2:62–66, 2009.
14. Zu-hua Shao. Efficient authenticated key agreement protocol using self-certified public keys from pairings. *Wuhan University Journal of Natural Sciences*, 10(1):267–270, 2005.
15. Yijuan Shi and Jianhua Li. Two-party authenticated key agreement in certificateless public key cryptography. *Wuhan University Journal of Natural Sciences*, 12(1):71–74, 2007.
16. Colleen M. Swanson. Security in key agreement: Two-party certificateless schemes. Master's thesis, University of Waterloo, Department of Combinatorics and Optimization, 2008.
17. Shengbao Wang and Zhenfu Cao. Escrow-free certificate-based authenticated key agreement protocol from pairings. *Wuhan University Journal of Natural Sciences*, 12(1):63–66, 2007.
18. Shengbao Wang, Zhenfu Cao, and Licheng Wang. Efficient certificateless authenticated key agreement protocol from pairings. *Wuhan University Journal of Natural Sciences*, 11(5):1278–1282, 2006.
19. Liang Xia, Shengbao Wang, Jiajun Shen, and Guoming Xu. Breaking and repairing the certificateless key agreement protocol from ASIAN 2006. *Wuhan University Journal of Natural Sciences*, 13(5):562–566, 2008.