On the Resilience of Key Agreement Protocols to Key Compromise Impersonation[⋆]

Maurizio Adriano Strangio

University of Rome “Tor Vergata”, ROME, ITALY
strangio@disp.uniroma2.it

Abstract. Key agreement protocols are a fundamental building block for ensuring authenticated and private communications between two parties over an insecure network. This paper focuses on key agreement protocols in the asymmetric authentication model, wherein parties hold a public/private key pair. In particular, we consider a type of known key attack called key compromise impersonation that may occur once the adversary has obtained the private key of an honest party. This attack represents a subtle threat that is often underestimated and difficult to counter. Several protocols are shown vulnerable to this attack despite their authors claiming the opposite. We also consider in more detail how three formal (complexity-theoretic based) models of distributed computing found in the literature cover such attacks.

Key words:key compromise impersonation, key agreement protocols

1 Introduction

Key agreement protocols are a fundamental building block for ensuring private communications between two parties over an insecure network (i.e. fully controlled by the adversary).

In general, (two-party) “authenticated key agreement” (AKE) protocols require entity authentication and key agreement to be appropriately linked in order to provide assurance that the session key is established only by the intended principals [12].

The property usually referred to as *implicit key authentication* (IKA) requires that in a run of the protocol uncorrupted principals are assured that no one aside from the intended partners in the communication can possibly learn the value of the session key. Sometimes a stronger property (*key confirmation*) is mandated that requires a party being assured that its intended partner has actually computed the session key (see [8, 5] for further discussion). Key confirmation is achieved at the expense of additional messages flows (e.g. using the compilers of [8, 7]); for this reason proving knowledge of the session key may be left to the subsequent communication (i.e. the calling applications).

When both IKA and key confirmation hold the protocol provides *explicit key authentication* (EKA).

[⋆] This is a revised version of the paper appearing in EuroPKI’06.

The whole point about key-based authentication, first introduced with the MTI/A0 [22] protocol, is that it allows for the design of efficient protocols with a reduced number of communication rounds and cryptographic operations. Although the messages exchanged are unauthenticated (i.e. there is no assurance they were sent by the intended partners in the communication and thus are perfectly "simulatable" by a third party), the IKA goal is achieved by letting the session key be "privately computable", i.e. requiring knowledge of long term private keying material known only by the principals running the protocol (assuming they are uncorrupted). Surprisingly, such protocols often enjoy important security properties (e.g. forward secrecy, key independence, etc).

Notice that key agreement protocols with the EKA property are not necessarily equivalent to generic AKE protocols since the later often have the message flows explicitly authenticated (e.g. with signatures and message authentication codes).

As usual, we designate the two generic parties participating in a protocol run as Alice and Bob. Now suppose an adversary (say Eve) has learned the private key of Alice either by compromising the machine running an instance of the protocol (e.g. with the private key stored in conventional memory as part of the current state) or perhaps by cloning Alice's smart card while she inadvertently left it unattended. Eve may now be able to mount the following attacks against the protocol:

1. impersonate Alice in a protocol run;
2. impersonate a different party (e.g. Bob) in a protocol run with Alice;
3. obtain previously generated session keys established in honest-party runs of the protocol.

In case 1. Eve can send messages on behalf of Alice and these will be accepted as authentic, in case 2. Eve could establish a session with Alice while masquerading as another party; this is known as Key Compromise Impersonation (KCI) and seems to appear for the first time in [18]. For example, Eve could impersonate a banking system and cause Alice to accept a predetermined session key and then obtain her credit card number over the resulting private communication link. In case 3. Eve may be able to decrypt the data exchanged by Alice and Bob in previous runs of the protocol (provided the transcripts are known). This is a weaker form of forward secrecy (which considers a passive adversary).

The preceding discussion demonstrates that long-term key compromise can lead to undesirable consequences at least until the corrupted principal discovers that his key was compromised. However, protocol designers are often concerned with forward secrecy and seem to ignore key compromise impersonation.

The main thesis of this paper is that key compromise impersonation is not less important than forward secrecy (as considered above); one should require that a secure key agreement protocol be also KCI-resilient since this security attribute is also related to party corruption.

In Section 4 we show that several implicitly authenticated key agreement protocols found in the literature do not withstand KCI attacks despite the authors claims. In order to offer a simplified and uniform treatment, all the protocols considered are specified in an elliptic curve setting since most of them were originally conceived in EC-based groups. In Section 5 we consider in more detail how three formal (complexity-theoretic based) models of distributed computing cover such attacks.

2 Notation and mathematical background

Given two strings s_1, s_2, the symbol $s_1 \| s_2$ denotes string concatenation. If X is a finite set then $x \overset{R}{\leftarrow} X$ denotes the sampling of an element uniformly at random from X. If α is neither an algorithm nor a set $x \leftarrow \alpha$ represents a simple assignment statement. The hash function $\mathcal{H}$ is used as a key derivation function (see [16] for practical KDFs).

The protocols we consider are based on EC cryptosystems. Let $\mathbb{F}_q$ denote the finite field containing q elements, where q is a prime power ($q = p$ or $q = 2^m$). An elliptic curve $\mathrm{E}(\mathbb{F}_q)$ over the field $\mathbb{F}_q$ (for simplicity we let $q = p$ with p a prime greater 3) is the set of points $P \equiv (P.x, P.y)$ that satisfy an (Weierstrass) equation of the form:

$$y^2 = x^3 + ax + b, \quad 4a^3 + 27b^2 \neq 0$$

where $a, b \in \mathbb{F}_q$. The set $\mathrm{E}(\mathbb{F}_q)$ with the operation of *point addition* $Q = P + R$ defined according to a *chord-and-tangent* rule and the point at infinity P_∞ serving as the identity element forms an (abelian) group structure. Repeated addition of a point P to itself x times is known as *scalar multiplication* $Q = xP$, this operation often dominates the execution time of elliptic curve based cryptographic schemes. The public elliptic curve domain parameters over $\mathbb{F}_q$ are specified by an 8-tuple:

$$\Phi_{EC} \equiv (q, FR, S, a, b, P, n, h)$$

where q is the underlying field order, FR (*field representation*) is an indication of the method used to represent field elements in $\mathbb{F}_q$, the *seed* S is for randomly generated elliptic curves, the two *coefficients* $a, b \in \mathbb{F}_q$ define the equation of the elliptic curve E over $\mathbb{F}_q$, the *base point* $P = (P.x, P.y)$ in $\mathrm{E}(\mathbb{F}_q)$, the prime order n of P and the cofactor $h = \sharp\mathrm{E}(\mathbb{F}_q)/n$. There are several well known algorithms for validating the parameters Φ_{EC} (see [13, 14]).

In an elliptic curve public-key cryptosystem user A is assigned a key pair (w_A, W_A) which is compatible with the set of domain parameters Φ_{EC}. In practice, the *private key* is a randomly selected value w_A in $[1, n-1]$ and the corresponding *public key* is the elliptic curve point W_A=$w_A P$.

There exist elliptic curve groups where the discrete logarithm assumption is known to hold: given the generator P and a random element $X \in \mathrm{E}(\mathbb{F}_q)$ it is computationally hard to compute $\log_P X$. More formally,

Assumption 1 (ECDL) *The EC group* $\mathrm{E}(\mathbb{F}_q)$ *satisfies the discrete logarithm assumption if for all PPT algorithms* $\mathcal{A}$ *we have:*

$$\Pr\left[x \overset{R}{\leftarrow} \mathbb{F}_q^*; X \leftarrow xP : \mathcal{A}(\Phi_{EC}, X){=}x\right] < \epsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of x) and ϵ is a negligible function.

The elliptic curve computational Diffie-Hellman (ECCDH) assumption holds in the group $\mathrm{E}(\mathbb{F}_q)$ if for random elements $X, Y \in \mathrm{E}(\mathbb{F}_q)$ it is computationally hard to compute $\log_P X \log_P Y P$ (i.e. if $X = xP$ and $Y = yP$ then the output should be xyP).

Assumption 2 (ECCDH) *The EC group* $\mathrm{E}(\mathbb{F}_q)$ *satisfies the computational Diffie-Hellman assumption if for all PPT algorithms we have:*

$$\Pr\left[x \xleftarrow{R} \mathbb{F}_q^*; y \xleftarrow{R} \mathbb{F}_q^*; X \leftarrow xP; Y \leftarrow yP : \mathcal{A}(\Phi_{EC}, X, Y){=}xyP\right] < \epsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ *(and random choices of* x, y*) and* ϵ *is a negligible function.*

3 A closer look at Key Compromise Impersonation

A KCI attack involves an adversary that has obtained the private key of an honest party. The adversarial goal is then to impersonate a different user and try to establish a valid session key with the "corrupted" party. This attack represents a serious and subtle threat since a user may not even be aware that his computer was "hijacked" and that a malicious party has obtained his private key. Set out below is a formal definition of KCI resilience:

Definition 1 (KCI-resilience) *A key agreement protocol is* KCI-resilient *if compromise of the long-term key of a specific principal does not allow the adversary to establish a session key with that principal by masquerading as a different principal.*

In the real world, a KCI attack is carried through as a man-in-the-middle attack. Let us consider the Unified Model [2, 15] IKA protocol described in Figure 1. Suppose adversary E knows w_A. Message Q_A is delivered to its intended recipient B without any interference from the adversary. Now, E intercepts B's response Q_B and replaces it with $Q_E = r_E P$. As a result, E causes A to accept the session key $\mathcal{H}(w_A W_B \| r_A Q_E)$ and is able to compute the same key as $\mathcal{H}(w_A W_B \| r_E Q_A)$. For this protocol, the attack works in exactly the same way if the adversary corrupts B.

Although the above example seems a trivial one, it is useful because it draws our attention on at least two important points: (1) many protocols are designed without considering KCI resilience as a security goal; (2) since messages are unauthenticated the corrupted party may not be able to detect the attack since a message received by the adversary (impersonating a legitimate user and deviating from the protocol specification) is perfectly indistinguishable (e.g. message Q_E above) from one received by an honest party.

We now turn to examine the MTI/A0 (Figure 2, [22]) and the MQV protocols (Figure 3, [20]) since they are apparently immune to KCI attacks. For both these implicitly authenticated key agreement protocols it appears to be infeasible to setup an attack (that exploits the algebraic group structure), similar to those presented in Section 4, with the only information known by the adversary being the long-term private key of a party. Indeed, for the MTI/A0 protocol Eve should be able to find a value Q_E such that the session key computed by A as $r_A W_B + w_A Q_E$ can also be calculated by an adversary knowing w_A. However, this does not seem possible unless either one of r_A or w_B are available to Eve. Similar reasoning also applies to the MQV protocol where the use of a non standard function[1] destroys the algebraic structure of the group. To be honest, resistance to KCI attacks was not the main design goal of the MQV protocol (and perhaps neither for the MTI/A0 protocol). In fact, the authors claim that the computation of $\overline{Q}$ (for a group element Q) was introduced for increased efficiency.

Notice that the protocols are easily broken if the adversary obtains the ephemeral data used by A, B (e.g. r_A, r_B or any other session-specific information stored in the current state); for this to occur, either the adversary is able to solve an instance of the discrete logarithm problem in an EC group (see Section 2) or she is given the capability of compromising a principals' machine (therefore obtaining the states of all running protocol instances at that time). The later case amounts to a stronger corruption model (which is also harder to put into practice) than the one considered in this paper.

$$
\begin{aligned}
A &: r_A \overset{R}{\leftarrow} [1, n-1] \\
&\quad Q_A \leftarrow r_A P \\
A \rightarrow B &: Q_A \\
B &: r_B \overset{R}{\leftarrow} [1, n-1] \\
&\quad Q_B \leftarrow r_B P \\
B \rightarrow A &: Q_B \\
A &: sk_A \leftarrow w_A W_B \| r_A Q_B \\
B &: sk_B \leftarrow w_B W_A \| r_B Q_A
\end{aligned}
$$

Fig. 1. Protocol UM

$$
\begin{aligned}
A &: r_A \overset{R}{\leftarrow} [1, n-1] \\
&\quad Q_A \leftarrow r_A P \\
A \rightarrow B &: Q_A \\
B &: r_B \overset{R}{\leftarrow} [1, n-1] \\
&\quad Q_B \leftarrow r_B P \\
B \rightarrow A &: Q_B \\
A &: sk_A \leftarrow \mathcal{H}(r_A W_B + w_A Q_B) \\
B &: sk_B \leftarrow \mathcal{H}(r_B W_A + w_B Q_A)
\end{aligned}
$$

Fig. 2. Protocol MTI/A0

4 Cryptanalysis of KCI-resilient AK Protocols

In this section we illustrate successful KCI attacks brought against some IKA key agreement protocols despite their authors have claimed resilience against such attacks. All the

[1] Recall that if Q ($\neq P_\infty$) is an elliptic curve point and $\overline{x}$ denotes the integer obtained from the binary representation of $Q.x$, then $\overline{Q}$ is the integer given by ($\overline{x}$ mod $2^{\lceil f/2 \rceil}$) $+ 2^{\lceil f/2 \rceil}$ where $f = \lfloor \log_2 n \rfloor + 1$.

$$\begin{array}{rl}
A: & r_A \overset{R}{\leftarrow} [1, n-1] \\
 & Q_A \leftarrow r_A P \\
A \rightarrow B: & Q_A \\
B: & r_B \overset{R}{\leftarrow} [1, n-1] \\
 & Q_B \leftarrow r_B P \\
B \rightarrow A: & Q_B \\
A: & s_A \leftarrow r_A + \overline{Q}_A w_A \\
 & sk_A \leftarrow s_A(Q_B + \overline{Q}_B W_B) \\
B: & s_B \leftarrow r_B + \overline{Q}_B w_B \\
 & sk_B \leftarrow s_B(Q_A + \overline{Q}_A W_A)
\end{array}$$

Fig. 3. Protocol MQV

protocols considered have optimal communication efficiency and computational complexity.

$$\begin{array}{rl}
A: & r_A \overset{R}{\leftarrow} [1, n-1] \\
 & Q_A \leftarrow r_A W_B \\
A \rightarrow B: & Q_A \\
B: & r_B \overset{R}{\leftarrow} [1, n-1] \\
 & R_B \leftarrow r_B W_A \\
 & T_B \leftarrow Q_A + R_B \\
 & Q_B \leftarrow R_B + r_B w_B^{-1} Q_A \\
B \rightarrow A: & Q_B \\
A: & T_A \leftarrow Q_A + w_A(w_A + r_A)^{-1} Q_B \\
 & sk_A \leftarrow T_A \\
B: & sk_B \leftarrow T_B
\end{array}$$

Fig. 4. Protocol LLK

4.1 LLK Protocol

The LLK key agreement protocol of Figure 4 is due to Lee *et al.* [21]. The session key is computed from the expression $(r_A w_B + r_B w_A)P$. Although the authors have conjectured that a KCI attack against the protocol is unfeasible, the following proves the opposite. Adversary E, impersonating B with knowledge of w_A, computes the EC point $Q_E \leftarrow r_E w_A^{-1}(Q_A + w_A W_B)$ and sends it to A in a run of the protocol. It is easily verified that A and E both accept with the key $\mathcal{H}(Q_A + r_E W_B)$.

4.2 SK Protocol

The SK key agreement protocol of Figure 5 was proposed by Song *et al.* [25]. The session key is derived from the expression $(r_A w_B + r_B w_A + r_A r_B)P$. The authors

$$
\begin{aligned}
A &: r_A \overset{R}{\leftarrow} [1, n-1] \\
& \quad Q_A \leftarrow r_A P \\
A \rightarrow B &: Q_A \\
B &: r_B \overset{R}{\leftarrow} [1, n-1] \\
& \quad Q_B \leftarrow r_B P \\
B \rightarrow A &: Q_B \\
A &: sk_A \leftarrow r_A W_B + (w_A + r_A) Q_B \\
B &: sk_B \leftarrow r_B W_A + (w_B + r_B) Q_A
\end{aligned}
$$

Fig. 5. Protocol SK

$$
\begin{aligned}
A &: r_A \overset{R}{\leftarrow} [1, n-1] \\
& \quad Q_A \leftarrow r_A W_B \\
A \rightarrow B &: Q_A \\
B &: r_B \overset{R}{\leftarrow} [1, n-1] \\
& \quad Q_B \leftarrow r_B W_A \\
B \rightarrow A &: Q_B \\
A &: sk_A \leftarrow r_A w_A^{-1} Q_B + w_A W_B \\
B &: sk_B \leftarrow r_B w_B^{-1} Q_A + w_B W_A
\end{aligned}
$$

Fig. 6. Protocol SSEB

claim resistance against KCI attacks. However, we now describe a successful KCI attack. Adversary E, impersonating B with knowledge of w_A, computes the EC point $Q_E \leftarrow r_E P - W_B$ and sends it to A in a run of the protocol. It is easily verified that A and E both accept with the key $\mathcal{H}(r_E(W_A + Q_A) - w_A W_B)$.

4.3 SSEB Protocol

The SSEB key agreement protocol of Figure 6 was designed by Al-Sultan *et al.* [1]. The session key is derived from the expression $(r_A r_B + w_A w_B)P$. The conjectured security attributes include KCI resilience. To prove that this claim is false it suffices for an adversary E, impersonating B with knowledge of w_A, to send the EC point $Q_E \leftarrow r_E w_A W_B$ to A in a run of the protocol. Once again it is easily verified that A and E both accept with the key $\mathcal{H}(r_E Q_A + w_A W_B)$.

4.4 PU Protocol

The PU key agreement protocol of Figure 7 was proposed by Popescu [23]. The conjectured security attributes include KCI resilience. The private/public key pair of a principal X is $(w_X, W_X = -w_X P)$ while the map $\mathcal{H}(\cdot)$ is a hash function. To communicate, principals A, B must share a static secret key $K_s = -w_A W_B = -w_B W_A = w_A w_B P$. A trivial attack shows that the protocol is not secure against KCI attacks. Indeed, an adversary E impersonating B with knowledge of w_A (and therefore can easily compute K_s), needs to send the EC point $Q_E \leftarrow r_E P$ to A in a run of the protocol. Clearly, both E and A will accept with the key $-r_E Q_A = -r_A Q_E$.

$$
\begin{aligned}
A &: r_A \overset{R}{\leftarrow} [1, n-1] \\
&\quad Q_A \leftarrow r_A P \\
&\quad e_A \leftarrow \mathcal{H}(Q_A.x, K_s.x) \\
A \rightarrow B &: Q_A, e_A \\
B &: r_B \overset{R}{\leftarrow} [1, n-1] \\
&\quad Q_B \leftarrow r_B P \\
&\quad e_B \leftarrow \mathcal{H}(Q_B.x, K_s.x) \\
B \rightarrow A &: Q_B, e_B \\
A &: e_B \overset{?}{=} \mathcal{H}(Q_B.x, K_s.x) \\
&\quad \text{if false then abort else } sk_A \leftarrow -r_A Q_B \\
B &: e_A \overset{?}{=} \mathcal{H}(Q_A.x, K_s.x) \\
&\quad \text{if false then abort else } sk_B \leftarrow -r_B Q_A
\end{aligned}
$$

Fig. 7. Protocol PU

4.5 Discussion

The implicitly authenticated protocols LLK, SSEB, SK, MTI use a similar approach to compute the session key. The intuition is that, in a protocol run, the established (fresh) session key results from an inextricable combination of information proving the identity of a principal (e.g. private keys w_A, w_B) and of session-specific data (e.g. r_A, r_B), in such a way to to avoid any danger of one of A or B being fooled into associating the key with the wrong principal. This is done by exploiting the algebraic structure of the underlying group. However, although this approach guarantees important security properties (e.g. key independence, forward secrecy), it makes possible for the adversary to mount the attacks illustrated above. As already pointed out before, to prevent these attacks, either the session key should be computed in such a way that the adversary is unable to cause the corrupted principal (say A) to accept a particular session key without the adversary knowing the private key of the principal she is impersonating (w_B) and/or the session-specific data of A (r_A). Alternatively, by following the approach used by the MQV protocol where such attacks are avoided by destroying the algebraic structure of the group.

The UM protocol and the PU protocol (and also the three protocols presented in [17]) are vulnerable to key compromise impersonation because they use a static shared key ($w_A W_B = w_B W_A$) to compute the session key. To obtain this key, it is sufficient for the adversary to corrupt either one of the principals A or B.

Adding key confirmation to the protocol may not necessarily help to counter KCI attacks since if the adversary has successfully caused the corrupted principal to accept a key then she knows the confirmation key too. To strengthen the protocol one could apply signatures to the message flows but then there would be no point in using IKA protocols at all, since there would be no real benefit with respect to plain Diffie-Hellmann key exchange authenticated with signatures.

Apparently, public key cryptography seems the only way to obtain KCI-resilient key agreement protocols.

5 KCI vs provable security

In general, it is difficult to formally prove that a protocol is KCI-resilient. Indeed, it is easier to find some specific message that demonstrates that the property does not hold at all (as shown in Section 4). To this end, it is worthwhile considering in more detail how the formal models of distributed computing found in the literature cover such attacks. We explore this issue in the present section. The focus is on three models:

– The model due to Bellare and Rogaway [5] was originally conceived for two-party protocols in the private-key authentication model. It was later extended to the password-based [4] and public-key settings [10, 9, 8]. This model introduces the notion of matching conversations as a means of entity authentication (and also partnering functions [6]). Secrecy of a session key is modeled by its indistinguishability from a random string;
– The approach followed by Shoup [24], for key agreement protocols in the asymmetric trust model, is completely different since it stems from the notion of simulatability (which is extensively present in the cryptographic literature as the basis for zero knowledge proofs);
– The initial version of the model of Bellare *et al.* [3] is also founded on the notion of simulatability. Later on, Canetti and Krawczyk [11] published a revised version with a reformulation of the security definitions in terms of the indistinguishability paradigm (a-la [5]).

Generally speaking, one expects that a formal security proof in the above models implies resilience to a whole range of attacks. However, although the adversarial models usually considered are almost all powerful (e.g. the adversary can relay messages out of order or to unintended recipients, concurrently interact with multiple instances of the protocol, corrupt legitimate users, acquire session keys, etc) a protocol is at best proven secure with respect to impersonation, known-key (Denning-Sacco) attacks and to some forms of long-term key exposure (e.g. forward secrecy).

5.1 KCI in the Bellare-Rogaway model

We briefly recall the main concepts of the Bellare-Rogaway [5] model in the asymmetric setting (with the extensions of [8, 9]). A (two-party) key agreement protocol is defined as a pair $\Sigma = (\mathcal{G}, \Pi)$ of poly-time computable functions where $\mathcal{G}$ is for generating the long-term (private-public) keys assigned to a principal P_i (suppose there is a finite number N of principals) and Π specifies the protocol actions and message formats. The symbol Π_i^r denotes the r-th protocol instance (oracle) run by principal P_i. Honest party oracles behave according to the description of the protocol Π. Oracle Π_i^r has an intended communication partner (say P_j) denoted by pid_i^r. The session identifier sid_i^r for the instance Π_i^r having $\mathsf{pid}_i^r = P_j$ is defined as the concatenation of messages transmitted to and received from the session Π_j^s (for some s). When $\mathsf{sid}_i^r = \mathsf{sid}_j^s$ we say that Π_i^r and Π_j^s have had a *matching conversation* (and the two communicating oracles are uniquely identified).

The adversary can initiate and interact with any (polynomial in ℓ — the security parameter) number of protocol instances (oracles) by asking the following queries:

– (init,i, j): this query sets $\mathsf{pid}_i^r = P_j$ and activates (the r-th) instance Π_i^r of the protocol at the principal P_i (in addition one may think of random coins being generated). As a result, oracle Π_i^r enters the idle state (i.e. waiting for a (send,$i, r,$ start) query);
– (send,i, r, M): the adversary sends message M to instance Π_i^r masquerading as pid_i^r. When $M \equiv$ start and instance Π_i^r is idle it responds with the first message according to the protocol specification and enters the expecting state. In all other cases the oracle moves into a state (e.g. continue, abort or accept) that depends on the initial state and on the message received;
– (execute,i, j): this query models a passive adversary eavesdropping on a run of the protocol between honest (uncorrupted) principals P_i, P_j. The resulting transcript is given to the adversary. In principle, an execute query can be simulated by send and init queries. However, in an execute query the parties and oracle instances strictly adhere to the protocol specification. This is opposed to send queries wherein the adversary can specify messages of her own choice;
– (reveal,i, r): this query models exposure of the session key of the instance Π_i^r due, for example, to improper erasure after its use, hijacking of the machine running the protocol or perhaps to cryptanalysis. It is applicable only to instances that have accepted;
– (corrupt,i): in the *weak corruption model* a corrupt query exposes the long-term private key of a principal P_i (as opposed to the *strong corruption model* wherein the adversary also obtains the internal state of the instances run by P_i). The adversary can use the compromised private key to impersonate P_i with send queries. We stress that the adversary does not obtain the session key as the result of a corrupt query on a instance Π_i^r that has accepted;
– (test,i, r): when the adversary $\mathcal{A}$ asks this query an unbiased coin b is flipped and K_b is returned. If $b = 0$ then $\mathsf{K}_0 \leftarrow \mathsf{sk}_i^r$ otherwise $\mathsf{K}_1 \overset{R}{\leftarrow} \{0,1\}^{\ell_1}$ (ℓ_1 is a secondary security parameter related to ℓ) and the adversary $\mathcal{A}$ must distinguish which one.

The security of protocol Σ is defined in the context of the following game between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$:

(a) Setup: The challenger $\mathcal{C}$ runs algorithm $\mathcal{G}(1^\ell)$ to generate private-public key pairs (SK_i, PK_i) for every principal P_i. The adversary is given the set $\{PK_i | i \in N\}$;
(b) Queries: Adversary $\mathcal{A}$ can adaptively ask (a polynomial in ℓ number of) oracle queries (a single test query is allowed). If required, both the challenger and the adversary can access a (public) random oracle modeling a hash function;
(c) Output: The adversary attempts to distinguish whether a key obtained from the test query is a real session key or a random one (or equivalently the adversary must output a correct guess b' of the bit b chosen by the challenger when answering the test query).

At the end of the above game the advantage of the adversary must be negligible for the protocol to be secure. In a concrete analysis this advantage is expressed as a function of the resource expenditure required to win the game.

To set out a meaningful notion of security the adversarial capabilities must be first specified; these are expressed in terms of the types of queries the adversary is allowed to ask during the game. For example, the weak form of forward secrecy, captures the inability of obtaining information on already generated session keys for a passive adversary that has corrupted the principals after a protocol run. This is modeled by leaving the adversary the ability to ask init, send, execute, reveal, corrupt queries in the game above. To win the game the adversary must try to guess bit b by asking the test query of a FS-fresh oracle, i.e. an oracle that (at the end of the game) has not been the target of a reveal query (neither has its partner oracle) and no send queries were asked of that oracle and its partner.

The advantage of the adversary is defined as $\mathsf{Adv}^{\mathsf{FS}}_{\Sigma}(\ell) = |2 \cdot \Pr[b' = b] - 1|$ and the protocol is FS-secure if the following inequality holds:

$$\mathsf{Adv}^{\mathsf{FS}}_{\Sigma}(\ell, t) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{FS}}_{\Sigma}(\ell)\} \leq \epsilon(\ell)$$

for negligible $\epsilon(\ell)$ and where the maximum is evaluated with respect to all adversaries running in polynomial time t (i.e. t is a polynomial in ℓ).

As it is, the model of Bellare-Rogaway offers no formalisation of KCI resilience. This is due to the corruption model used that allows only a restricted notion of oracle freshness which captures weak forward secrecy (as defined before). Indeed, the basic notion of a fresh oracle does not allow the adversary to corrupt the principal running the target session nor the principal running the partner session, and the adversary is not allowed to ask reveal queries neither of the target session nor of any matching session.

In order to provide for such a possibility, we present the following definition of a KCI-fresh oracle. The adversary can ask the test query to a KCI-fresh oracle in the game defined above while being able to ask init, send, reveal, corrupt queries.

Definition 2 (KCI-fresh oracle) *An oracle Π^r_i is* KCI-fresh *if the following conditions hold at the end of the game:*

1. $\mathsf{acc}^r_i = \mathsf{TRUE}$*;*
2. *neither* (reveal, i, r) *nor* $(\mathsf{corrupt}, j)$ *queries were asked by the adversary;*
3. *if the adversary has queried* $(\mathsf{corrupt}, i)$*, then no* (send, j, s, M) *query was asked, where M is a message chosen by the adversary and* $\mathsf{pid}^s_j = P_i$*.*

The advantage of the adversary is defined as $\mathsf{Adv}^{\mathsf{KCI}}_{\Sigma}(\ell, t) = |\Pr[b' = b] - \frac{1}{2}|$ and it must be negligible for the protocol to be KCI-secure.

The above definition of a KCI-fresh oracle (obviously) requires that the private key of the corrupted principal P_i was not used by the adversary to impersonate P_i to other principals. Observe also that Π^r_i may not terminate with a partnered oracle (if such an oracle exists then we should also require that no reveal queries were asked of that oracle).

Note that in the strong corruption model we would require that the adversary learned no session state information of the target oracle thus implying that the corruption occurred before any sessions were initiated at that party.

Unfortunately, key compromise impersonation resilience must be established on its own since there appears to be no relationship, for example, with (weak) forward secrecy (which, on the other hand, can be shown to imply key independence). However, a protocol secure against key compromise impersonation also maintains (weak) partial forward secrecy (wpFS), i.e. a passive adversary cannot learn any information on the session key of the target oracle even after obtaining the private key of a specific principal. More formally, the definition of a wpFS-fresh oracle is simply obtained from Definition 2 under the assumption that the test oracle Π_i^r and its partner, say Π_j^s (for some s), have had a matching conversation.

Definition 3 (wpFS-fresh oracle) *An oracle* Π_i^r *is* wpFS-fresh *if the following conditions hold at the end of the game:*

1. $\mathsf{acc}_i^r = \mathsf{acc}_j^s = \mathsf{TRUE}$ *and* $\mathsf{sid}_i^r = \mathsf{sid}_j^s \neq \mathtt{NULL}$*;*
2. (reveal, i, r)*,* (reveal, j, s) *queries were not asked by the adversary;*
3. *if the adversary has queried* $(\mathsf{corrupt}, i)$ *or* $(\mathsf{corrupt}, j)$*, then no* (send, j, s, M) *nor* $(\mathsf{send}, i, r, M')$ *queries was asked, where* M, M' *are messages chosen by the adversary.*

We now prove the following theorem.

Theorem 1 *Given the* EC *parameters* Φ_{EC}*, the* MTI/A0 *protocol (Figure 2) is a* KCI*-resilient protocol assuming the group* $E(\mathbb{F}_q)$ *satisfies the* ECCDH *assumption and the hash function* $\mathcal{H}$ *is modeled as a random oracle. Concretely, we have*

$$\mathsf{Adv}^{\mathsf{KCI\text{-}R}}_{\mathrm{MTI/A0}}(\ell, t, q_h, q_{re}, q_{co}, q_{se}) \leq 1/N^2 \cdot 1/q_h \cdot \epsilon,$$

where t is the total running time of the game played by the adversary (including its execution time), ℓ *the security parameter and* $q_h, q_{co}, q_{re}, q_{se}$*, respectively, the number of random oracle,* corrupt*,* reveal *and* send *queries and* N *is the number of principals.*

Proof. Given $X = xP, Y = yP$ the symbol DH(X, Y) denotes the Diffie-Hellman secret xyP. The proof is by a reduction technique; if an adversary $\mathcal{A}$ is able to break KCI-resilience then we may construct an adversary $\mathcal{F}$ that uses $\mathcal{A}$ as a subroutine and succeeds in solving the computational Diffie-Hellman problem (CDHP) in the underlying elliptic curve group E$(\mathbb{F}_q)$. Algorithm $\mathcal{F}$ simulates the game played by $\mathcal{A}$ (against the challenger $\mathcal{C}$ — see above) in such a way that $\mathcal{A}$'s view is indistinguishable from the real game. The description of $\mathcal{F}$ follows:

1. $\mathcal{F}$ receives in input ($X = xP$, $Y = yP$), chooses i^*, j^* (we assume that $i^* \neq j^*$) guessing that i^* will be the principal corrupted by $\mathcal{A}$ in its game and that j^* is the principal impersonated by $\mathcal{A}$ in the attack;
2. $\mathcal{F}$ generates private/public keys (w_i,W_i) for all principals P_i where $i = j^*$ and sets $W_{j^*} = Y$
3. $\mathcal{F}$ runs $\mathcal{A}$ as a subroutine answering its queries as follows:
   - For (send,i, r, M) queries, when Π_i^r is in the idle state the answer is aP for random a (with oracle Π_i^r moving into the expecting state if M = start). If $i = i^*$ and $\mathsf{pid}_{i^*}^r = P_{j^*}$ then the response is $aP + X$;
   - For (send,i, r, M) queries, when Π_i^r is in the expecting state the session key sk_i^r is set to a random element in $\{0, 1\}^\ell$ (and oracle Π_i^r moves into the accepting state); if $i = i^*$, $\mathsf{pid}_{i^*}^r = P_{j^*}$ (with $\mathsf{sid}_{i^*}^r$ prefixed by $aP + X$) and a (corrupt,i^*) query was asked $\mathcal{F}$ stores the record ($aP + X$,M) in the list L1;

- For random oracle queries $\mathcal{H}(i, j, U, V, W)$ the response is a random element sampled from $\{0, 1\}^\ell$ (or the value output previously for the same argument); if $i = i^*$ and $\mathsf{pid}_{i^*}^r = P_{j^*}$ then $\mathcal{F}$ finds the record $(aP + X, M)$ in L1 (if it exists) such that $U = aP + X$ and $V = M$ and writes $(aP + X, M, W)$ to list L2;
- init, execute, reveal, test queries are answered normally;
- (corrupt,i) queries are answered as usual except that if $i = j^*$ then $\mathcal{F}$ aborts.

4. When $\mathcal{F}$ terminates (exactly when $\mathcal{A}$ does) it chooses a random element in the list L2 and outputs DH(X, Y)=$W - aY - w_{i^*}M$ (where $W = (a + x)Y + w_{i^*}M$). Observe that oracles $\Pi_{i^*}^u$, for some u, such that $\mathsf{sid}_i^u = U\|V$ are KCI-fresh according to the simulation (and therefore any test query that $\mathcal{A}$ asks of these oracles can be correctly answered by $\mathcal{F}$).

It is straightforward to verify that the success probability of $\mathcal{F}$ is bounded from above by $1/N^2 \cdot 1/q_h \cdot \epsilon$. □

5.2 KCI in the Canetti-Krawczyk model

Recently, Krawczyk [19] has formally defined KCI attacks in the model of Canetti-Krawczyk [11]. The formalism is introduced to prove the resilience of a hash-based version of the MQV protocol (HMQV). Two communication models are considered; the first one is the simplified authenticated-links (AM) model wherein the communication links are assumed to be authenticated, the second one is the unauthenticated-links (UM) model wherein the network is totally under the control of the adversary. In both the models the adversary is given capabilities which allow different levels of information exposure of a session and/or principal (the adversary may ask queries session-state reveal, party-corruption, session-key query, session expiration, test-session).

A secure key exchange protocol is formalised ([11], definition 4) in a context similar to the game of Section 5.1 by requiring that (1) if two uncorrupted principals complete matching sessions then (with overwhelming probability) they both output the same key and (2) the probability of success of the adversary in distinguishing the session key from a random one is no greater than 1/2 plus a negligible function.

In the model it is hypothesized that in real world implementations long-term secret keys are often granted better protection (e.g. by using cryptographic modules) than session-specific data; this is reflected in the attackers' capabilities by considering separate party corruption and session state reveal operations. The authors speculate that whenever this is not a realistic assumption one could weaken the model by omitting the session-state reveal operation. However, in practice almost *all* computations can take place in a cryptographic module (e.g. those involving the generation of ephemeral Diffie-Hellman public keys) thus making session-specific information leakage more difficult. Furthermore, hardware-specific attacks (e.g. power analysis) are not considered.

The basic definition of a secure protocol does not look upon the case of corrupted principals, therefore, in [19] a new notion is introduced into the model to account for KCI attacks, namely, that of a clean session. The goal is to capture the situations wherein the adversary has learned the long-term private key of a principal but has not actively controlled the session (e.g. by impersonating the principal) during a run of the protocol.

A key agreement protocol is considered resilient to KCI attacks if the adversary is unable to distinguish the real session key (from a random one) of a complete session, being this session clean and the peer session (if it exists) at an uncorrupted principal also clean. Under this definition ([19], Definition 20) it is shown that the HMQV protocol is secure in the model of [11].

5.3 KCI in the Shoup model

In the formal model of Shoup [24] security is defined via simulation. There is an ideal world wherein the service offered by the key agreement protocol is defined, and a real world which describes how protocol participants communicate. An ideal world adversary is essentially constrained to be benign. A security proof shows that the adversary in the real world is essentially "simulatable" by the adversary in the ideal world and therefore one deduces that the protocol is secure in the real world. Again, the simulation takes place in the context of a game similar to those defined in the preceding models. Three classes of adversaries are defined, according to their capability of obtaining private information held by users (either static or ephemeral data), that give rise to static corruptions, adaptive corruptions and strong adaptive corruptions.

Let us examine how KCI attacks can be viewed in the adaptive corruptions model (in the static corruptions model the adversary holding the private key of a principal may simply decide to actively impersonate that principal in a protocol run). We use the notation and terminology of [24]. Consider an instance I_{ij} engaging in the key agreement protocol (e.g. the two pass LLK protocol) with a *compatible* instance $I_{i'j'}$. Suppose that after the first message M_1 (e.g. $Q_i = r_i W_{i'}$ in protocol LLK) is delivered, $I_{i'j'}$ accepts the session key $K_{i'j'}$. The adversary now corrupts user U_i. Instance $I_{i'j'}$ responds with a message M_2 (e.g. $Q_{i'} = r_{i'} W_i$ in protocol LLK), the adversary intercepts it and instead delivers message $\overline{M}_2$ (e.g. $\overline{M}_2 \equiv \overline{Q}_{i'} = \overline{r}_{i'} w_i^{-1}(Q_i + w_i W_{i'})$ in protocol LLK). At this point I_{ij} will accept a session key K_{ij} known by the adversary and different from $K_{i'j'}$ (which the adversary ignores). Now, in the ideal world, when $I_{i'j'}$ generated its session key, it was not corrupted so the only connection assignment possible for $I_{i'j'}$ is create. On the other hand, the only possible connection assignment for I_{ij}, being U_i corrupted, is compromise. However, I_{ij} and $I_{i'j'}$ are *compatible*, hence I_{ij} cannot be compromised without breaking the rules of the game since $PID_{ij} = ID_{i'}$ is assigned to user $U_{i'}$. Moreover, a connect is also not possible between I_{ij} and $I_{i'j'}$ since this would imply $K_{ij} = K_{i'j'}$. We must conclude that the simulation is not possible since it would lead to inconsistent real world and ideal world transcripts. Note that we have used the *liberal compromise rule* as defined in [24] (the simulation is still not be possible under the *conservative compromise rule*).

6 Conclusions and future work

In this paper we discussed key compromise impersonation resilience for key agreement protocols in the asymmetric trust model. Several protocols, whose authors have mistakenly claimed resilience to KCI, are proven vulnerable to such attacks. For these protocols, explicit key confirmation (e.g. using the compilers of [8, 7]) may provide an effective countermeasure since the parties involved (A, B) accept different session keys. However, this is achieved at the expense of increased computational and round complexity.

It appears that protocol designers do not always pay attention to key compromise impersonation. Instead, forward secrecy, which is indeed another harmful threat related to party corruption, is usually considered more important. However, our thesis is that the security analysis of key agreement protocols is incomplete with a corruption model that considers only forward secrecy.

Although there is a constant debate in the research community concerning formal (complexity-theoretic based) security models, they undoubtedly constitute a valuable approach to achieve proactively secure key agreement protocols. Surprisingly, however, three of the most significant models found in the literature do not have a satisfactory approach (besides having one at all) to KCI. We have attempted to incorporate a reasonable notion of KCI resilience into the model of Bellare-Rogaway. Future work includes formulating an appropriate notion of resilience to KCI into the formal security model of Shoup.

References

1. K. Al-Sultan, M. Saeb, M. Elmessiery, and U.A.Badawi. A new two-pass key agreement protocol. *Proceedings of the IEEE Midwest 2003 Symp. on Circuits, Systems and Computers*, 2003.
2. R. Ankney, D. Hohnson, and M. Matyas. The Unified Model. *Contribution to X9F1*, 1995.
3. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. *In 30th Symposium on Theory of Computing*, pages 419–428, 1998.
4. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attack. *In Proceedings of EUROCRYPT 2000*, LNCS 1807:139–155, 2000.
5. M. Bellare and P. Rogaway. Entity authentication and key distribution. *In Proceedings of CRYPTO 1993*, LNCS 773:232–249, 1993.
6. M. Bellare and P. Rogaway. Provably secure session key distribution - the three party case. *In Proceedings of 27th ACM Symposium on the Theory of Computing 1995*, 1995.
7. M. Bellare and P. Rogaway. The AuthA protocol for password-based authenticated key exchange. *Contribution to IEEE P1363*, 2000.
8. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. *In Proceedings of the 6th IMA Int.l Conf on Cryptography and Coding*, LNCS 1355:30–45, 1997.
9. S. Blake-Wilson and A. Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. *Security Protocols - 5th International Workshop*, LNCS 1361:137–158, 1998.
10. S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellmann key agreement protocols. *Selected Areas in Cryptography - 5th International Workshop*, LNCS 1556:339–361, 1999.
11. R. Canetti and H. Krawczyk. Analysis of key exchange protocols and their use for building secure channels. *Advances in Cryptology-EUROCRYPT 2001*, LNCS 2045:453–474, 2001.
12. W. Diffie, P. van Oorschot, and M. Wiener. Authentication and Authenticated Key Exchange. *Designs, Codes and Cryptography*, 2:107–125, 1992.
13. FIPS-PUB-186-2. Digital Signature Standard. National Institute of Standards and Technology, 2000.

14. D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Professional Computing, New York, 2004.
15. IEEE-P1363-2000. Standard specifications for public key cryptography. Institute of Electrical and Electronics Engineers, 2000.
16. IEEE-P1363.2/D15. Standard specifications for password-based public key cryptographic techniques. Institute of Electrical and Electronics Engineers, 2004.
17. I. Jeong, J. Katz, and D. Lee. One-Round Protocols for Two-Party Authenticated Key Exchange. *Applied Cryptography and Network Security 2004*, 2004.
18. M. Just and S. Vaudenay. Authenticated Multi-Party Key Agreement. *Advances in Cryptology-ASIACRYPT 1996*, LNCS 1163:36–49, 1996.
19. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellmann protocol. *http://eprint.iacr.org/2005/176*, 2005.
20. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28:119–134, 2003.
21. C. Lee, J. Lim, and J. Kim. An efficient and secure key agreement. *IEEE p1363a draft*, 1998.
22. T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *Transactions of IEICE*, VolE69:99–106, 1986.
23. C. Popescu. A Secure Authenticated Key Agreement Protocol. *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference*, 2004.
24. V. Shoup. On Formal Models for Secure Key Exchange. Technical Report RZ 3120, IBM Research, 1999.
25. B. Song and K. Kim. Two-pass authenticated key agreement protocol with key confirmation. *Progress in Cryptology - Indocrypt 2000*, LNCS 1977:237–249, 2000.