On Forward Secrecy in One-Round Key Exchange

Colin Boyd and Juan González Nieto

Information Security Institute
Queensland University of Technology
Brisbane, Australia
{c.boyd|j.gonzaleznieto}@qut.edu.au

Abstract. Most one-round key exchange protocols provide only weak forward secrecy at best. Furthermore, one-round protocols with strong forward secrecy often break badly when faced with an adversary who can obtain ephemeral keys. We provide a characterisation of how strong forward secrecy can be achieved in one-round key exchange. Moreover, we show that protocols exist which provide strong forward secrecy and remain secure with weak forward secrecy even when the adversary is allowed to obtain ephemeral keys. We provide a compiler to achieve this for any existing secure protocol with weak forward secrecy.

1 Introduction

Two-party key agreement continues to be a topic of active research. Although we understand much better now than we did ten years ago how to design such protocols and how to model their security, controversy remains as to precisely what is the right security model and new designs, security proofs and attacks continue to be published [19, 8, 11]. Moreover, there are continuing trends to find new useful security properties, model them and provide instantiating protocols.

Forward secrecy. One of the most important security properties for key agreement is *forward secrecy*. This is the property that compromise of long-term keys should not result in compromise of session keys agreed before the long-term keys were compromised. This property was first articulated by Diffie, van Oorschot and Wiener [14] and has been incorporated into various recent formal security models [2, 17]. Forward secrecy is often regarded as an optional extra to fundamental security properties, particularly confidentiality and implicit authentication of the session key. However, it is always a desirable property if it can be provided without too much computational overhead.

Generally speaking, protocols which provide forward secrecy are computationally more expensive than those which dispense with it. It is also often the case that forward secrecy requires more communications complexity, in that more messages are sent between the two parties of the protocol. A typical way of providing forward secrecy is to ensure that the session key is defined only

in terms of *ephemeral* secrets while the messages exchanged are authenticated using long-term secrets. The classic Diffie–Hellman key exchange [13] is a typical component for the ephemeral exchange, while signatures can be used for authentication. Since it is necessary to prevent replayed messages, each party typically provides a random challenge or *nonce* which is signed together with the ephemeral Diffie–Hellman value. This means that the complete protocol uses three messages.

As well as the *number* of messages, a useful measure of the communications efficiency is the number of protocol *rounds*. One protocol round consists of all messages which can be communicated simultaneously. In recent years it has become commonplace to present two-party key exchange protocols in one round versions [16–18, 26, 15]. This means that the two protocol messages can be sent independently and simultaneously.

Weak forward secrecy. In 2005, Krawczyk presented the, now well-known, protocol HMQV [17] as a one-round two-message protocol (there is also a one-message variant). Krawczyk pointed out that HMQV does not provide forward secrecy in its full generality. Indeed, he showed an explicit attack in the case that the adversary is allowed to be *active* in the session which it is attacking. This means that the adversary can manipulate messages sent to the target victim; later, when the victim has completed the session and erased the session key, the adversary corrupts the victim and is able to reconstruct the session key. This attack applies to other protocols too, in particular those in which the messages exchanged are the same as those in basic Diffie–Hellman exchange. However, the attack does not apply to the many protocols which use the sender's long-term key to construct the message. In Section 2 we will present a variation of Krawczyk's attack which applies to a very broad class of one-round protocols.

Krawczyk defined a restricted form of forward secrecy which assumes that the adversary is not active during the session under attack. This certainly covers a wide variety of attack scenarios but is not as comprehensive a threat model as we would like to have. Following Krawczyk [17] we will say that a protocol achieves *weak forward secrecy* (wFS) if it provides forward secrecy when the adversary is not active in the test session. To clearly distinguish it from the ideal security level, in this paper we say that a protocol achieves *strong forward secrecy* (sFS) if forward secrecy is provided even when the adversary is active in the test session. It seems reasonable to assume that in some scenarios an adversary would be active against a chosen session with the intention of later attempting to gain access to long-term keys of its victims so sFS is arguably an important property.

Definition 1 (Strong and weak forward secrecy (informal)). *A protocol provides forward secrecy if the adversary cannot distinguish the agreed session key from a random string even given the long-term keys of both parties after the session is complete. There are two types of forward secrecy.*

- *If this property still holds even if the adversary takes an active part in the session under attack we say that the protocol provides* strong forward se-

crecy. *In this case the victim executes the session with the adversary who masquerades as a legitimate party.*
- *If this property only holds when the adversary is prevented from taking an active part in the session under attack we say that the protocol provides* weak forward secrecy. *In this case the victim executes the session with a legitimate party whose messages are transmitted correctly to the victim.*

Related work. Krawczyk pointed out that many one-round protocols only achieve weak forward secrecy. Indeed, he stated that weak forward secrecy is the best possible for "... *any* 2-message key-exchange protocol ... with no secure shared state previously established between the parties" [17, page 15]. This seems to have led many researchers to believe incorrectly that sFS is always impossible for one-round key exchange. For example, LaMacchia *et al.* [18] have stated that Krawczyk's argument showed that "no 2-round AKE protocol can achieve full perfect forward secrecy (PFS)" while Boyd *et al.* [5] say "Krawczyk showed that forward secrecy in the usual sense cannot be achieved in a two-pass protocol such as the ones that we consider." Chow and Choo [9] stated that "two-party protocols with only two-message flow and having no previous establishment of secure shared state cannot achieve perfect forward secrecy". Saar *et al.* [23] said "no two-pass key exchange protocol can achieve forward secrecy".

Nevertheless, one-round protocols with strong forward secrecy do exist. In 2010 Gennaro *et al.* [15] have shown that a much older one-round protocol of Okamoto and Tanaka, with minor modifications, does indeed provide strong forward secrecy. A protocol of Jeong, Katz and Lee [16] from 2004 is another example. It is a natural question to ask exactly when sFS can be achieved. Gennaro *et al.* point out that sFS is impossible for a one-round protocol when "information transmitted between the parties is computed without access to the parties' long-term secrets"[1].

Although it is not often acknowledged, Bellare, Pointcheval and Rogaway [2, Section 7] seem to have been the first to define a notion of weak forward secrecy for key exchange. Their definition is strongly related to that given by Krawczyk, but the model of Bellare *et al.* [2] has some significant differences from that used by Krawczyk. Specifically, they do not separate an adversary's ability to obtain long-term keys from the ability to obtain session state (including ephemeral keys). This means that their definition of wFS is different and although they point out that sFS (in their definition) is not possible for two-message protocols, their simple attack in such a scenario would not be valid in models popular today (see Section 2.1 for further discussion on models).

Recently, in independent work, Cremers and Feltz [10] have proposed a new one-round protocol which provides strong forward secrecy (and also deniability). For their security proof, rather than forbid all ephemeral keys to be revealed by the adversary, they define a new model which precisely prevents the adversary from obtaining useful ephemeral keys. The result is to ensure that the attack

[1] Gennaro *et al.* call this situation *implicit authentication* but we avoid this terminology here because it conflicts with other usages of the same term.

we present in Section 2.2 is not available to the adversary in their model. The construction of their protocol can be seen as related to the compiler we construct in this paper, the main difference being that they use signatures where we use a MAC. The consequence of this is that our construction can be more efficient, but in compensation the protocol of Cremers and Feltz provides security against a stronger adversary.

Our results. Our first result in this paper is to show that one-round key exchange protocols with sFS are *impossible* if the adversary is allowed to reveal the ephemeral secrets of the protocol. Specifically we show that sFS is not possible when the adversary is allowed to reveal ephemeral secrets of the partner party to the test session. We achieve this by defining a very general class of two-party one-round key exchange protocols and presenting a variation of Krawczyk's attack which applies to all such protocols. This result gives a clear separation between models where sFS is achievable and those where it is not.

As implied by our first result, the models used by Gennaro *et al.* [15] and by Jeong *et al.* [16] to prove sFS for their protocols have to disallow ephemeral key reveal queries to the adversary. In some application scenarios this may be a reasonable assumption, but suppose that an adversary is later found to be able to obtain ephemeral secrets after all. Of course sFS is no longer possible, but what about wFS or other security properties? It turns out that both of these protocols break very badly in this situation - not only is wFS not obtained when ephemeral keys can be revealed, but the protocols are no longer secure in any meaningful sense. Thus these two protocols are sensitive to changes in the security model and there seems no reason why this should have to occur. The second main result of this paper is to show that more robust security is possible. Specifically we demonstrate existence of protocols which provide sFS when ephemeral key compromise is forbidden to the adversary, but maintain wFS and basic protocol security if ephemeral keys do become available to the adversary. As well as strengthening our theoretical understanding of security models for key exchange, we believe that this result points out a practically important way to achieve robust security for real-world protocols.

Paper outline. In the next section we review definitions of strong and weak forward secrecy in modern security models for key exchange. We then explain why strong forward secrecy cannot be obtained when ephemeral key reveal is available to the adversary, on the normal assumption that replays cannot be detected without interaction. Section 3 illustrates that current protocols either fail to achieve strong forward secrecy, even under a weakened adversary, or fail to achieve weak forward secrecy under a strong adversary. This leads us to propose a compiler to convert weakly secure protocols into ones which have sFS. Applying this compiler we can obtain many protocols which provide strong forward secrecy under a weakened adversary but which degrade gracefully to provide weak forward secrecy when faced with a strong adversary. Finally we propose some avenues for further research.

2 Strong forward secrecy in one round

In this section we will describe in what circumstances forward secrecy can be achieved in terms of the security model in use. We first need to establish how forward secrecy is captured in currently used computational models.

2.1 Defining forward secrecy

Cryptographic models for key exchange were initiated by Bellare and Rogaway in the 1990s [3, 4]. These initial papers did not include a definition of forward secrecy, but later Pointcheval joined the authors to extend their model to deal with various properties [2] including forward secrecy. In recent years an alternative model by Canetti and Krawczyk [1, 6] has become more popular; we will use an extended version of this model due to Krawczyk [17] later in this paper and we will refer to this as the CK model.

The CK model uses the same basic idea as the Bellare and Rogaway model. The adversary is an efficient (PPT) algorithm which controls communications between the set of protocol participants. A session is an instance of a protocol at a specific party; each session has an associated session state which is distinct from the long-term keys of the party which owns the session. The adversary activates sessions and sends messages to sessions which respond according to the protocol specification. At some point a session can complete and accept a session key. The session identifier of a completed session s is (A, B, Out, In) where A is the owner of the session, B is the intended peer party, and Out, In are the messages sent and received by the session. A different completed session s' is said to be a matching session of s if its session identifier is (B, A, In, Out). Note that a matching session may not exist even when s completes; indeed this corresponds exactly to the situation when the adversary is active in s.

The adversary is allowed access to various oracles which correspond to plausible capabilities of real-life adversaries, as follows.

session key reveal This query takes as input a session identifier and returns the session key if the session is complete.

session state reveal This query takes as input a session identifier. If the session is incomplete then the session state is returned. This can typically be ephemeral keys.

corruption This query takes as input a party identifier and returns all the keys of the party. This includes long-term keys, session state and session key (if available).

expiry This query takes as input a session identifier and deletes the session key (and any related session state). While it has no output, expiry is of major importance in defining strong forward secrecy.

At some point during its run the adversary has to nominate a completed test session. The adversary's goal is then to distinguish the session key from a random string. In response to the test query the adversary is given one of two

strings each with probability 1/2; one string is the session key from the test session and the other is a key drawn uniformly at random from the session key space. The adversary may then continue with other queries before outputting a bit b identifying whether or not the string is the session key. The adversary's advantage is the probability that it chooses the correct value of b minus 1/2.

The adversary is not allowed to make queries which would trivially allow it to obtain the session key. Specifically, no session key reveal is allowed against the test session or its matching session (if it exists). In addition, no corruption is allowed at the test session or its matching session (if it exists) before the session is expired. However, in order to model strong forward secrecy we allow the adversary to corrupt the owner of the test session and the peer party *after* the session has expired. We also disallow session state reveal on the test session and its matching session (if it exists). Note, however, that when the adversary is active in the test session it chooses the messages received by the session and can choose its own ephemeral values.

Definition 2 ([17]). *A key-exchange protocol is called* CK-secure *if for all adversaries $\mathcal{A}$:*

1. *if two uncorrupted parties complete matching sessions in a run of the protocol then, except for a negligible probability, the session key output in these sessions is the same;*
2. *$\mathcal{A}$ succeeds (in its test-session distinguishing attack) with probability not more that 1/2 plus a negligible fraction.*

Notice that protocols which satisfy this formal definition of security automatically have strong forward secrecy. This is because the adversary is allowed to expire the test session and subsequently corrupt the parties to the test session. Furthermore, there is no restriction on the adversary to be active during the test session. Canetti and Krawczyk [6] extended their definition to protocols without any forward secrecy by removing the expire query from the adversary's list of available oracles. Later, Krawczyk [17] formally defined weak forward secrecy by disallowing the adversary from constructing its own messages during the test session — messages must simply be relayed between uncorrupted parties for the test session.

Cremers [11] has pointed out that matching in the CK model is not defined for sessions which are not complete which means that we cannot always be sure whether the adversary is allowed state reveal queries against incomplete sessions. This will not cause us a problem in the subsequent proof since we will show that state reveal can never be allowed when strong forward secrecy is required.

In 2007, LaMacchia *et al.* [18] introduced a new model known as eCK intended for analysis of two-party key agreement protocols based on Diffie–Hellman. We will not define the eCK model in detail here. Instead we point out some of its main features relevant to forward secrecy. A comparison of the differences between the eCK and CK models has been made by other authors [11, 25].

A main aim of the eCK model is to capture a wider range of attacks than the CK model by allowing exposure of ephemeral keys and long-term keys of

the test session and its peer. Instead of considering session state, the eCK model assumes that there will be an ephemeral key generated by each party and the model defines an ephemeral key reveal query. Since the session key can be trivially computed given ephemeral and long-term keys of one party to the test session, exposure of both ephemeral and long-term keys from the same session is forbidden. However other exposures are allowed, in particular the adversary can obtain the ephemeral key of the test session and the long-term key of the peer party. This is not possible in the CK model since session-state reveal is explicitly forbidden for the test session and so this has led to the notion that the eCK model is more powerful than the CK model. However this is not the case as was later observed by several authors [5, 25, 11]. Cremers [11] has conducted a careful comparison of the CK model (in the original and modified versions) and the eCK model and shown clearly that they cannot be ordered according to strength.

Note that it is impossible to fully describe strong forward secrecy in the eCK model because there is no notion of session expiry in the eCK model. Indeed the model forbids exposure of the long-term key of the peer in the case that no matching session of the test session exists (which is the same as saying that the adversary is active in the test session). However, protocols proven secure in the eCK model provide weak forward secrecy since the session key remains secure even if both long-term keys involved in the session are exposed.

2.2 Strong forward secrecy and ephemeral key reveal

We first present a generic definition of one-round two-party key exchange to allow us to describe an abstract result. We assume that any user A holds a long-term secret x_A which is used for all protocol instances and generates a new short-term ephemeral secret r_A for each protocol instance. There is also public information associated with the protocol. Common protocols based on Diffie–Hellman key exchange expect users to hold public keys of the form $y_A = g^{x_A}$ in some suitable group generated by g, but we do not assume this structure. Public information will typically include user public keys, user identities, and other system parameters such as the algebraic setting and master public keys.

In a one-round protocol, users can only send a single message and this message must be independent of the message sent by the other party. In general the message sent by A is generated by some function f, randomised by some random input r_A from some set R and f may have other inputs including x_A and any public information. We denote the output of $f(\cdot)$ by t_A. Note that t_A may have some arbitrary structure (particular some distinct components) in general. On receiving the message t_B from its peer, user A can then compute the shared secret Z_{AB} using some other function which we denote $F(\cdot)$. The inputs to F are t_B and any of the same inputs used for f. Note, however, that F must be deterministic if A and B are to compute the same value Z_{AB}. The peer of A will compute Z_{AB} in a symmetrical way but it is not necessary that the computation is identical. One example is that the computation may depend on the ordering of the peer identities in some natural ordering. Therefore we describe functions $\hat{f}$ and $\hat{F}$ used by B, analogous to f and F used by A – for most published

protocols $f = \hat{f}$ and $F = \hat{F}$. Protocol 1 illustrates the generic protocol messages and shared secret computation.

A		B
$r_A \leftarrow R$		
$t_A = f(x_A, r_A, \text{public info})$	$\xrightarrow{t_A}$	$r_B \leftarrow R$
		$t_B = \hat{f}(x_B, r_B, \text{public info})$
$Z_{AB} = F(x_A, r_A, t_B, \text{public info})$	$\xleftarrow{t_B}$	$Z_{AB} = \hat{F}(x_B, r_B, t_A, \text{public info})$

Protocol 1: Generic one-round key agreement protocol

To our knowledge all prominent two-party key exchange protocols satisfy our generic description. In many protocols, such as HMQV [17], the messages t_A and t_B are simply Diffie–Hellman messages: $t_A = g^{r_A}$ and $t_B = g^{r_B}$ where g generates a suitable group. Other protocols, such as Naxos [18], include the long-term key of the sender in the computation of t_A. All the protocols we consider rely on some variant of the Diffie–Hellman problem: finding g^{xy} given g^x and g^y. Of particular relevance in this paper is the *gap Diffie–Hellman* (GDH) assumption [22] which says that solving the (computational) Diffie–Hellman problem is hard even given access to a decisional oracle for the Diffie–Hellman problem.

We are now in a position to describe how the adversary $\mathcal{A}$ can break the forward secrecy of the generic protocol given access to ephemeral keys. In this analysis we need to explicitly assume that the parties are not able to detect replay of a valid message from the peer party. This is not an unusual assumption since both messages have to be independent of fresh input from the peer party in any one-round protocol. Indeed, this assumption holds for all prominent one-round protocols [17, 18, 26]. Protocols using timestamps or counters to detect replays would not be vulnerable to this attack, but most modern protocols prefer to avoid relying on these mechanisms. The attack of adversary $\mathcal{A}$ proceeds as follows.

1. $\mathcal{A}$ observes a protocol instance between A and B and records the first message t_A from A. $\mathcal{A}$ asks for the ephemeral value r_A used in this instance.
2. $\mathcal{A}$ initiates a new instance of the protocol with B. In this instance $\mathcal{A}$ masquerades as A. $\mathcal{A}$ sends t_A from the earlier protocol instance as the first message of the new instance. B will compute a new r_B and t_B and send t_B to A which is captured by $\mathcal{A}$.
3. $\mathcal{A}$ eventually expires the session and corrupts A to obtain x_A. $\mathcal{A}$ now has all the inputs of F so can re-compute Z_{AB}.

The impact of this attack is that the adversary can recover any information sent by B during the session established in step 2.

Proposition 1. *A one-round key exchange protocol cannot provide strong forward secrecy in any model which allows the adversary to reveal ephemeral secrets*

of the partner party to the test session. This holds even if ephemeral secrets cannot be revealed during the test session.

The attack process described above is a variant of the attack which Krawczyk uses to illustrate that HMQV cannot achieve strong forward secrecy. The main difference is that Krawczyk does not include step 1 of our attack. Since HMQV has messages which are independent of the long-term secret x_A, the active adversary can start the attack at step 2 and generate a valid message with a new random r_A value. In order to avoid Krawczyk's attack, any one-round protocol with strong forward secrecy must have messages which depend on the long-term private key of the sender. However, this alone is not sufficient. For example, the Naxos protocol which we examine below includes long-term private keys of the sender in the messages but does not provide strong forward secrecy.

3 Reconciling strong and weak forward secrecy

Having clarified when sFS cannot be achieved, in this section we turn to consideration of when sFS *can* be achieved. We first take a look at protocols with a proof of sFS and note that they all seem to have severe drawbacks against powerful adversaries[2]. We also look at some well-known protocols secure against strong adversaries and note that these do not provide sFS even when ephemeral keys cannot be revealed. This leads us to look for protocols with a more graceful degradation of security.

3.1 Current protocols with sFS

Most recent one-round key exchange protocols provide only wFS. Two exceptions are the mOT protocol of Gennaro *et al.* [15] and protocols TS2/TS3 of Jeong, Katz and Lee [16]. As we know must be the case from the results in Section 2, the models used by Gennaro *et al.* and Jeong *et al.* for their security proofs do not allow ephemeral keys to be revealed. The mOT protocol refines an early identity-based protocol of Okamoto and Tanaka [20, 21] but can be easily converted to a traditional public-key version. If an adversary is allowed to reveal ephemeral keys the mOT protocol is completely insecure – the adversary can immediately obtain the sender's long-term private key as a consequence.

Protocol TS3 of Jeong *et al.* is shown as Protocol 2. This protocol, which we will refer to as the JKL protocol, is essentially basic Diffie–Hellman authenticated by a message authentication code (MAC) whose key is the static Diffie–Hellman between A and B. The properties of Protocol 2 are similar to signed Diffie–Hellman but use of the MAC gives better efficiency. (We note that a one-round version of signed Diffie–Hellman was proven secure by Shoup [24] in a model with static corruption.)

[2] An exception is the recent protocol of Cremers and Feltz [10] mentioned in the introduction.

Shared information: Static key, $S_{AB} = g^{x_A, x_B}$.

A		B
$r_A \leftarrow R$		
$t_A = g^{r_A}$	$\xrightarrow{t_A, \mathsf{MAC}_{S_{AB}}(A, B, t_A)}$	$r_B \leftarrow R$
		$t_B = g^{r_B}$
$Z_{AB} = t_B^{r_A}$	$\xleftarrow{t_B, \mathsf{MAC}_{S_{AB}}(B, A, t_B)}$	$Z_{AB} = t_A^{r_B}$

Protocol 2: Jeong–Katz–Lee (JKL) protocol TS3

Consider an adversary $\mathcal{A}$ against Protocol 2 who can obtain ephemeral keys. We already know that sFS cannot be achieved, but unfortunately things are worse than that. Once $\mathcal{A}$ has obtained the value r_A for a protocol run of A with B, $\mathcal{A}$ can replay the first message of the protocol against B as many times as he likes and always obtain the session key. Therefore this protocol, just like mOT, does not even provide a basic level of security once ephemeral keys may be revealed.

3.2 Remaining secure with a more powerful adversary

Security proofs for key exchange are usually only provided in a single model. This is fine when we understand the security context in which the protocol is to be deployed. In practice it is quite likely that a protocol, particularly a standardised one, will be deployed in a variety of contexts. Moreover, the security context may change after the protocol is deployed. It therefore seems a very desirable property that a protocol should degrade its security level gracefully when faced with a stronger adversary than originally envisaged. From this viewpoint the mOT and JKL protocols both behave very badly. We would therefore like to find whether there is a protocol which provides sFS when ephemeral key reveal is forbidden yet still provides wFS when ephemeral key reveal becomes available to the adversary.

Since current protocols with sFS do not provide what we are looking for, it is natural to try starting from a protocol which is already known to provide wFS against a strong adversary and see if it can achieve sFS when ephemeral key reveal is forbidden. One obvious candidate protocol is the Naxos protocol [18] shown as Protocol 3. Parties A and B hold long-term secret keys x_A and x_B respectively, with corresponding public keys $y_A = g^{x_A}$ and $y_B = g^{x_B}$. LaMacchia *et al.* [18] gave a proof of the Naxos protocol on the assumption that the gap Diffie–Hellman problem is hard. Hash functions H_1 and H_2 are modelled as random oracles in the proof.

Unfortunately it turns out that Naxos does not provide sFS even when ephemeral key reveal is forbidden (to prevent the generic attack in Section 2.2).

A		B
$r_A \leftarrow R$		
$h_A = H_1(x_A, r_A)$		
$t_A = g^{h_A}$	$\xrightarrow{t_A}$	$r_B \leftarrow R$
		$h_B = H_1(x_B, r_B)$
		$t_B = g^{h_B}$
$Z_{AB} = H_2(t_B^{x_A}, y_B^{h_A}, t_B^{h_A}, A, B)$	$\xleftarrow{t_B}$	$Z_{AB} = H_2(y_A^{h_B}, t_A^{x_B}, t_A^{h_B}, A, B)$

Protocol 3: Naxos protocol

Given the long-term key of B, an active adversary $\mathcal{A}$ can obtain the session key from an expired session as follows.

1. $\mathcal{A}$ chooses a random ephemeral private key r'_B and sends $t'_B = g^{r'_B}$ to party A.
2. A will compute the shared secret as $Z_{AB} = H_2({t'_B}^{x_A}, y_B^{h_A}, {t'_B}^{h_A}, A, B)$.
3. After session expiry $\mathcal{A}$ can also compute $Z_{AB} = H_2(y_A^{r'_B}, t_A^{x_B}, t_A^{r'_B}, A, B)$.

This allows $\mathcal{A}$ easily to compute the session key once the long-term private key of B, x_B, is revealed. It turns out that a similar 'attack' is possible on CMQV [26] too. We therefore need to look for a new example of a protocol which provides graceful security degradation. In the next section we will provide a generic way to provide many such examples by designing a compiler which takes any protocol secure in the eCK or CK models and converts it into a protocol with sFS.

Table 1 summarises what we have found with regard to existing prominent protocols and also lists our new variant of the Naxos protocol shown as Protocol 5 in Section 4. Security analysis of Protocol 5 will be presented in Section 4. For each protocol we show the security status in the two cases where (i) the adversary is allowed to obtain the ephemeral keys (for the eCK model) or session state information (for the CK model) and (ii) when this is not allowed. Recall that we cannot obtain sFS in the normal eCK model but it potentially could be achieved if ephemeral key reveal is disallowed and long-term keys for the test session are kept private until after session expiry. However, we know that HMQV, Naxos and CMQV do not achieve sFS even under these conditions or in the CK model without state reveal queries.

4 A compiler for protocols with graceful security degradation

In this section we present a compiler, denoted $\mathcal{C}$, based on the JKL protocol which transforms a protocol secure in either the CK or eCK model into one which provides sFS in a model without ephemeral key reveal. The idea is to add

the authentication method used in the JKL protocol using an independent key. Thus any party A now has two independent public keys, y_A with corresponding private key x_A, and $y'_A = g^{x'_A}$. The original protocol will be authenticated using a MAC keyed with the static key $S_{AB} = g^{x'_A x'_B}$.

As in the JKL protocol [16] we require that the MAC is strongly unforgeable under chosen message attack in the standard definition. Furthermore, the assumption that the MAC key is a group element can be adapted to suit real-world MACs by applying a randomness extractor as also discussed by Jeong *et al.* [16].

Protocol 4 shows the result of applying the compiler $\mathcal{C}$ to the generic Protocol 1 from Section 2.2. The additional cost of the compiled protocol is just one exponentiation, one MAC generation and one MAC verification per party.

Private keys of A : x_A, x'_A
Private keys of B : x_B, x'_B
Shared information: Generator g of group G. Static key, $S_{AB} = g^{x'_A x'_B}$.

A		B
$r_A \leftarrow R$		
$t_A = f(x_A, r_A, \text{public info})$	$\xrightarrow{t_A, \mathsf{MAC}_{S_{AB}}(A, B, t_A)}$	$r_B \leftarrow R$
		$t_B = \hat{f}(x_B, r_B, \text{public info})$
	$\xleftarrow{t_B, \mathsf{MAC}_{S_{AB}}(B, A, t_B)}$	
$Z_{AB} = F(x_A, r_A, t_B, \text{public info})$		$Z_{AB} = \hat{F}(x_B, r_B, t_A, \text{public info})$

Protocol 4: Image of Protocol 1 under compiler $\mathcal{C}$

In order to make a formal security claim we need to adapt the CK security model. By definition of forward secrecy we must allow the adversary to obtain the long-term keys of the parties in the test session. In addition we must allow the adversary to be active in the test session so it is essential to prevent the adversary from obtaining the long-term keys until after the test session is complete. Therefore we should allow the adversary access to the Expiry query and then allow corruption of the test session after it is expired, which will reveal only

Table 1. Comparison of protocol security with and without ephemeral key/state reveal

Protocol	Ephemeral key/state reveal? Yes	No
HMQV [17]	wFS (CK)	wFS
mOT [15], JKL [16]	Insecure	sFS
Naxos [18], CMQV [26]	wFS (eCK)	wFS
Protocol 5	wFS (eCK)	sFS

the long-term keys. Furthermore, in order to thwart the attack of Section 2.2, we must disallow session state reveal queries to the partner party to the test session (in addition to disallowing session state reveal to the test session itself). Therefore the model we use can be defined to be the CK model, as defined in Section 2.1, without these session state reveal queries; we will refer to this model as the CK-NSR model.

Theorem 1. *Suppose π is a protocol which is secure in either the eCK model or in the CK model with wFS. Then $\mathcal{C}(\pi)$ is secure in the CK-NSR model with sFS, assuming that the DDH problem in G is hard and the MAC is a secure message authentication code. Moreover, $\mathcal{C}(\pi)$ remains secure with wFS in the CK/eCK model (whichever model π is secure in).*

Proof. We first claim that since the original protocol π already has a security proof in either the CK or the eCK model it is straightforward to see that the security proof will still hold for $\mathcal{C}(\pi)$ since the private keys used for the MAC scheme are generated independently of the long-term keys used for the computations of the original protocol π. A trivial reduction will allow a successful adversary against protocol $\mathcal{C}(\pi)$ to be converted into a successful adversary against π.

Now we turn to the more interesting case of showing that $\mathcal{C}(\pi)$ provides sFS. We consider a sequence of security games: Game_0 to Game_3. Denote by S_i the event that an adversary $\mathcal{A}$ against Game_i succeeds.

$Game_0$: This is the original sFS game between an adversary $\mathcal{A}$ in the CK-NSR model and protocol $\mathcal{C}(\pi)$. The advantage of an adversary against Game_0 is defined as

$$Adv_{\mathcal{C}(\pi)} = \left|\Pr(S_0) - \frac{1}{2}\right|. \tag{1}$$

$Game_1$: This game is the same as Game_0 except that the game aborts if $\mathcal{A}$ sends a valid message to any session which was not previously output by another uncorrupted session.

Let $\mathcal{A}$ be an adversary against $\mathcal{C}(\pi)$ and let forgery be the event that the $\mathcal{A}$ forges a valid message. Then,

$$|\Pr(S_1) - \Pr(S_0)| \leq \Pr(\text{forgery}). \tag{2}$$

In order to bound the probability of a forgery, we construct an adversary $\mathcal{F}$ against the MAC scheme. $\mathcal{F}$ simulates the environment for the adversary $\mathcal{A}$ in Game_1 and will attempt to forge a MAC in the event that forgery occurs.

The input to $\mathcal{F}$ consists of the parameters of the MAC scheme, which includes access to a MAC oracle. Suppose that n_u is a bound on the number of users of the protocol. For all users $\mathcal{F}$ generates a key pair for the MAC keys (the x'_I). $\mathcal{F}$ generates the other long-term private keys (the x_I) for all users. $\mathcal{F}$ picks a pair of identities A, B at random from the set of all users of the protocol.

$\mathcal{F}$ then starts adversary $\mathcal{A}$. Note that $\mathcal{F}$ knows all the private keys and therefore it can perfectly simulate the interface expected by $\mathcal{A}$ for all parties. However,

we modify the simulation as follows: for sessions between A and B, instead of using the long term MAC secret $g^{x'_A x'_B}$, $\mathcal{F}$ will make use of the MAC oracle that it has access to in the MAC security game that $\mathcal{F}$ is simultaneously playing. Further, in the case that $\mathcal{A}$ corrupts A or B, $\mathcal{F}$ aborts the protocol. Recall that we have defined the key to the MAC to be an element of the group generated by g. Then, from the DDH assumption, one can see that this simulation is indistinguishable from perfect.

Eventually the event $\mathsf{forgery}$ occurs which includes a valid MAC of a protocol user. With probability at least $1/n_u^2$ this is a MAC of user A in a session with B. When this occurs neither A nor B can have been corrupted so the protocol has not been aborted. Denote by $\mathsf{success}_{\mathcal{F}}$ the probability that $\mathcal{F}$ succeeds in breaking the unforgeability of the MAC scheme. Then we have shown the following, where Adv_{DDH} is the advantage of an efficient adversary against the DDH problem and Adv_{MAC} is the advantage of an efficient adversary in breaking the MAC scheme.

$$\Pr(\mathsf{forgery}) \leq n_u^2(Adv_{DDH} + Adv_{MAC}). \tag{3}$$

Game$_2$: This game is the same as Game$_1$ except for the following. Two users A and B are chosen at random amongst all users. If $\mathcal{A}$ chooses any other two parties for the test session, or does not choose the test session to belong to user A, then the game aborts.

A and B represent a guess at the parties in the test session selected by $\mathcal{A}$, specifically that the test session belongs to user A with peer party B. This is a game transition based on a large failure event in the sense of Dent [12]. Since the probability that the guess of both parties is correct is bounded by $1/n_u^2$ we have:

$$\Pr(S_2) = \frac{1}{n_u^2}\left(\Pr(S_1) - \frac{1}{2}\right) + \frac{1}{2}. \tag{4}$$

Game$_3$: This game is the same as Game$_2$ except for the following. A value n^* is chosen at random with $1 \leq n^* \leq n_s$ where n_s is the maximum number of session which $\mathcal{A}$ is allowed to schedule between any two parties. We denote the n^*-th session between A and B by s^*. The session s^* represents a guess at the test session which will be selected by $\mathcal{A}$. This is also a transition based on a large failure event so that:

$$\Pr(S_3) = \frac{1}{n_s}\left(\Pr(S_2) - \frac{1}{2}\right) + \frac{1}{2}. \tag{5}$$

In order to bound the probability of success of adversary $\mathcal{A}$ against Game$_3$, we now construct an adversary $\mathcal{B}$ against protocol π that makes use of $\mathcal{A}$ in the event that it does not abort. We show that in this case $\mathcal{B}$ can win its game whenever $\mathcal{A}$ wins. If Adv_π is the advantage of an efficient adversary against π we thus obtain:

$$\Pr(S_3) \leq \frac{1}{2} + Adv_\pi. \tag{6}$$

Lemma 1. *There is an efficient adversary $\mathcal{B}$ against protocol π which wins exactly when $\mathcal{A}$ wins Game$_3$ against $\mathcal{C}(\pi)$.*

Proof (of Lemma 1). We construct an explicit adversary $\mathcal{B}$. The input to $\mathcal{B}$ consists of the public keys and parameters for π. $\mathcal{B}$ interacts with protocol π using queries as per the underlying security model (CK or eCK) being considered. The task of $\mathcal{B}$ is to distinguish the session key in its test session from a random key. Although the adversary $\mathcal{A}$ is allowed to be active in the test session, the effect of including the MAC is that only messages formed by the simulator $\mathcal{B}$ will be valid as determined by Game$_1$. This makes $\mathcal{A}$ a passive adversary in all sessions except those where it replays messages formed by $\mathcal{B}$ in earlier sessions. $\mathcal{B}$ chooses independent random key pairs $(x'_I, g^{x'_I})$ for each party I and sends the public key components together with the parameters and keys it received to $\mathcal{A}$.

$\mathcal{B}$ now simulates all the required oracles for $\mathcal{A}$, making a one to one correspondence with every session run by $\mathcal{A}$, except for sessions between A and B. That is, all queries that $\mathcal{A}$ makes to a party in a session that does not involve both A and B are simulated by simply relaying the same query to the corresponding oracles available to $\mathcal{B}$ in the $\mathcal{C}(\pi)$ game that $\mathcal{B}$ is playing. The reason why the simulation for sessions between A and B needs to be different is that in the test session of the π game $\mathcal{B}$ must be passive, whereas in the $\mathcal{C}(\pi)$ game $\mathcal{A}$ can inject old messages from B to A.

In Game$_3$ we know that $\mathcal{A}$ chooses the specific session s^* as its test session. $\mathcal{A}$ is free to choose either a fresh message or replay an old message which will be sent to party A, but $\mathcal{B}$ needs to ensure that it uses a fresh message. Therefore before session s^* occurs, whenever $\mathcal{A}$ asks for a fresh message to be sent from B to A, $\mathcal{B}$ asks its challenger for a fresh message m and returns this to $\mathcal{A}$. If $\mathcal{A}$ asks for the response to this message m from A then $\mathcal{B}$ also asks for the response r_1 from its challenger but stores r_1 and then replays the message m and obtains a new response r_2, which it forwards to $\mathcal{A}$. In this way $\mathcal{B}$ is free to choose the session with m and r_1 as its test session at A in the case that $\mathcal{A}$ replays message m in session s^*.

For the session s^*, $\mathcal{B}$ waits to see whether $\mathcal{A}$ invokes a fresh message from B to A or replays an old message. If it is a fresh message then $\mathcal{B}$ can invoke a fresh message from its challenger and choose this session as its own test session. If it is a replay then $\mathcal{B}$ sends the response that it stored the first time that the message was invoked from its challenger and chooses that session as its own test session. Either way the test session for both $\mathcal{A}$ and $\mathcal{B}$ has the same messages. $\mathcal{B}$ can therefore forward its challenge key K to $\mathcal{A}$ and send $\mathcal{A}$'s response at its own response. Note also the $\mathcal{B}$ can continue to simulate sessions after session s^* using the oracles from its own challenger. Thus $\mathcal{B}$ wins exactly if $\mathcal{A}$ wins. □

Combining (1)-(6), we get

$$Adv_{\mathcal{C}(\pi)} \leq n_u^2 n_s \left(Adv_{DDH} + Adv_{MAC} + Adv_{\pi}\right). \tag{7}$$

Since Adv_{DDH}, Adv_{MAC} and Adv_{π} are all negligible by assumption, then so is $Adv_{\mathcal{C}(\pi)}$. It is easy to see that all the reductions described above run in polynomial time, which completes the proof of security of $\mathcal{C}(\pi)$. □

Protocol 5 shows the result of applying the compiler $\mathcal{C}$ to the Naxos protocol (Protocol 3). In contrast to the protocols discussed in Section 3.1, this protocol provides sFS but remains secure, even with wFS, when ephemeral key reveal is allowed. Indeed Protocol 5 is still secure in the eCK model. More efficient protocols with these properties are certainly possible since we can apply $\mathcal{C}$ to more efficient existing protocols such as CMQV [26].

Private keys of A : x_A, x'_A
Private keys of B : x_B, x'_B
Shared information: Generator g of group G. Static key, $S_{AB} = g^{x'_A x'_B}$.

A		B
$r_A \leftarrow R$		
$h_A = H_1(x_A, r_A)$		
$t_A = g^{h_A}$	$\xrightarrow{t_A, \mathsf{MAC}_{S_{AB}}(A,B,t_A)}$	$r_B \leftarrow R$
		$h_B = H_1(x_B, r_B)$
		$t_B = g^{h_B}$
$Z_{AB} = H_2(t_B^{x_A}, y_B^{h_A}, t_B^{h_A}, A, B)$	$\xleftarrow{t_B, \mathsf{MAC}_{S_{AB}}(B,A,t_B)}$	$Z_{AB} = H_2(y_A^{h_B}, t_A^{x_B}, t_A^{h_B}, A, B)$

Protocol 5: Application of compiler $\mathcal{C}$ to Naxos protocol

5 Discussion

We have clarified the situations when strong forward secrecy can and cannot occur for one-round key exchange protocols. Specifically we have shown that sFS can never be achieved (with normal assumptions) if the adversary is allowed to reveal ephemeral secrets. We have also shown a generic method of achieving sFS at the cost of adding roughly one exponentiation per user to the cost of an existing secure protocol. This allowed us to construct protocols which provides sFS when ephemeral keys cannot be revealed and still maintain security with wFS when ephemeral keys can be revealed.

There are some obvious open questions that arise from this work.

- The idea of graceful security degradation can be explored for other properties such as key compromise impersonation. Generally, considering what happens to protocols under different security models seems to be unexplored territory.
- It is not obvious that the cost of one exponentiation is the minimum required to promote secure protocols to ones with sFS. Are there more efficient protocols which can provide graceful degradation from sFS to wFS when a stronger adversary is encountered?

– It could be interesting to consider whether the picture is any different in the universal composability model. In this paper we have concentrated on the popular computational models. Some equivalence results do exist [7]. In our view it is important to get the situation clear in the well-used models first.

Acknowledgements

We are grateful to Cas Cremers and Michèle Feltz for illuminating discussions regarding forward secrecy, generic attacks and models. Anonymous referees made some helpful comments; in particular a referee from PKC 2011 provided very accurate corrections.

References

1. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *30th ACM Symposium on Theory of Computing*, pages 419–428. ACM Press, 1998. Full version at `http://www-cse.ucsd.edu/users/mihir/papers/key-distribution.html`.
2. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, pages 139–155. Springer-Verlag, 2000. Lecture Notes in Computer Science Volume 1807.
3. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology – Crypto '93*, pages 232–249. Springer-Verlag, 1993. Lecture Notes in Computer Science Volume 773. Full version at `http://www-cse.ucsd.edu/users/mihir`.
4. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution – the three party case. In *27th ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, 1995.
5. Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. Efficient one-round key exchange in the standard model. In Yi Mu et al., editors, *Information Security and Privacy, 13th Australasian Conference, ACISP 2008*, volume 5107 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2008.
6. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt 2001*, pages 453–474. Springer-Verlag, 2001. Lecture Notes in Computer Science Volume 2045. `http://eprint.iacr.org/2001/040/`.
7. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In L. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2002.
8. Sanjit Chatterjee, Alfred Menezes, and Berkant Ustaoglu. Reusing static keys in key agreement protocols. In Bimal K. Roy and Nicolas Sendrier, editors, *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2009.
9. Sherman S. M. Chow and Kim-Kwang Raymond Choo. Strongly-secure identity-based key agreement and anonymous extension. In *Information Security*, pages 203–220. Springer-Verlag, 2007. Full version: `http://eprint.iacr.org/2007/018`.

10. Cas Cremers and Michèle Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300, 2011. http://eprint.iacr.org/.
11. Cas J.F. Cremers. Formally and practically relating the CK, CK-HMQV, and eCK security models for authenticated key exchange. Cryptology ePrint Archive, Report 2009/253, 2009. http://eprint.iacr.org/.
12. Alexander W. Dent. A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260, 2006. http://eprint.iacr.org/2006/260/.
13. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
14. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchange. *Designs, Codes and Cryptography*, 2:107–125, 1992.
15. Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Okamoto-Tanaka revisited: Fully authenticated Diffie-Hellman with minimal overhead. In Jianying Zhou and Moti Yung, editors, *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010*, volume 6123 of *Lecture Notes in Computer Science*, pages 309–328, 2010.
16. I.R. Jeong, J. Katz, and D.H. Lee. One-round protocols for two-party authenticated key exchange. In *Applied Cryptography and Network Security*, pages 220–232. Springer, 2004.
17. H. Krawczyk. HMQV: A high-performance secure Diffie–Hellman protocol (extended version available from http://eprint.iacr.org/2005/176/). In *Advances in Cryptology – Crypto 2005*, pages 546–566. Springer-Verlag, 2005. LNCS.
18. Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In W. Susilo et al., editors, *Provable Security, First International Conference, ProvSec 2007*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2007.
19. Alfred Menezes and Berkant Ustaoglu. Comparing the pre- and post-specified peer models for key agreement. *IJACT*, 1(3):236–250, 2009.
20. Eiji Okamoto. Key distribution systems based on identification information. In C. Pomerance, editor, *Advances in Cryptology – Crypto '87*, pages 194–202. Springer-Verlag, 1987. Lecture Notes in Computer Science Volume 293.
21. Eiji Okamoto and Kazue Tanaka. Key distribution system based on identification information. *IEEE Journal on Selected Areas in Communications*, 7(4):481–485, May 1989.
22. Tatsuaki Okamoto and David Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In K. Kim, editor, *International Workshop on Practice and Theory in Public Key Cryptography (PKC 2001)*, Lecture Notes in Computer Science, pages 104–118.
23. Augustin P. Sarr, Philippe Elbaz-Vincent, and Jean-Claude Bajard. A new security model for authenticated key agreement. Cryptology ePrint Archive, Report 2010/237, 2010. http://eprint.iacr.org/.
24. Victor Shoup. On formal models for secure key exchange. http://www.shoup.net, November 1999.
25. B. Ustaoglu. Comparing SessionStateReveal and EphemeralKeyReveal for Diffie–Hellman protocols. *Provable Security*, pages 183–197, 2009.
26. Berkant Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Des. Codes Cryptography*, 46(3):329–342, 2008.