On Post-Compromise Security

Katriel Cohn-Gordon, Cas Cremers and Luke Garratt
Department of Computer Science, University of Oxford
first.last@cs.ox.ac.uk
February 29, 2016

Abstract

In this work we study communication with a party whose secrets have already been compromised. At first sight, it may seem impossible to provide any type of security in this scenario. However, under some conditions, practically relevant guarantees can still be achieved. We call such guarantees "post-compromise security".

We provide the first informal and formal definitions for post-compromise security, and show that it can be achieved in several scenarios. At a technical level, we instantiate our informal definitions in the setting of authenticated key exchange (AKE) protocols, and develop two new strong security models for two different threat models. We show that both of these security models can be satisfied, by proposing two concrete protocol constructions and proving they are secure in the models. Our work leads to crucial insights on how post-compromise security can (and cannot) be achieved, paving the way for applications in other domains.

1. Introduction

If all of a party's secrets are compromised by an adversary, is there any hope of securely communicating with them in the future? The folklore answer is "no": a party's long-term secret key *defines* their identity, and anyone knowing an identity key can impersonate its owner. This applies broadly to protocols with different security models, including authenticated key exchange and messaging. In all cases, traditional definitions of protocol security rule out compromise-and-impersonate attacks as indefensible.

These scenarios are a pressing practical concern; modern secure systems are designed assuming that some compromise might eventually occur, and aim instead to limit its scope. For example, devices frequently fall temporarily under adversarial control: later-removed malware, short-term physical access, and confiscation at a border crossing all describe scenarios where a potential adversary has access to a device which is then returned to the original owner. In the traditional paradigm, we cannot make security guarantees about these *post-compromise* cases. Our central thesis is that PCS is not just theoretically possible, but in fact practically implementable and already influencing the design of state-of-the-art protocols. We describe PCS precisely, explain the limits of what can be achieved, and give formal security models that capture it.

Definition 1 (informal)**.** *A protocol between Alice and Bob provides* Post-Compromise Security *(PCS) if Alice has a security guarantee about communication with Bob, even if Bob's secrets have already been compromised.*

The concept of PCS fills multiple gaps in the theoretical literature. In practice, various cryptographic schemes aim to provide an informal version of such guarantees. However, without a formal analysis there are many unanswered questions: To what extent can we allow compromise? What must schemes do to achieve PCS? In what sense are some schemes fundamentally limited in their ability to achieve PCS? How does PCS apply to real-world protocols used today? We go about answering these questions, defining models for security after weak and total compromise.

Weak compromise corresponds to temporary adversarial control of long-term key operations, without actual theft of the long-term key. Such a situation can occur when an adversary has temporary access to a party's hardware security module (HSM). A HSM is a physical computing device that safeguards long-term keys, providing cryptographic operations such as signing without allowing direct access to the key. It is commonly believed that if an adversary has only temporary access to a HSM but not the long-term key itself, then once this access is revoked security can still be achieved. We concretely define this notion and show that such guarantees can indeed be proven formally. However, the application of a HSM to achieve PCS is non-trivial; we comment in particular that a recent TLS 1.3 proposal [22] is not post-compromise secure.

Total compromise is more severe, and corresponds to an adversary who learns all the keys of a party. The folklore that PCS is unachievable in this case is backed up by an informal argument, proceeding roughly as follows. An adversary learning the long-term key of some party ("Bob") can run the same computation that Bob would perform, using the compromised key and generating new random numbers if needed. Since this is exactly what Bob would do, nobody can distinguish between the true key owner and the adversary's impersonation of them.

This informal proof of impossibility does not consider the case where the previously compromised agent later completes a subsequent session. Indeed, such a session might generate a new and uncompromised secret. If this secret is then used for authentication, the original compromise may not be sufficient to mount the impersonation attack.

Katriel Cohn-Gordon is supported by the EPSRC, through the Oxford Centre for Doctoral Training in Cyber Security.

Some protocols already maintain "key continuity" in this fashion. Examples include the telephony protocol ZRTP [9] and the instant messaging protocols OTR [5] and its asynchronous successor Axolotl [25]; the latter has been implemented in Signal (and, partially, WhatsApp[1]) and Langley's Pond [24]. These protocols all implement a key rotation scheme, whereby each communication key is used to generate a successor and then deleted. For this case, we formally define a mechanism to synchronise and rotate keys between communicating parties, and show that it achieves PCS even after a full compromise of the long-term key. However, as we will show, there are subtleties involved where this can go wrong.

Contributions. We make the following contributions.

1) We introduce the high-level notion of *post-compromise security*. This notion has wide applicability, and we use it to unify the potential security guarantees offered by HSMs and by regular key rotation.
2) We instantiate post-compromise security in the context of authenticated key exchange (AKE), advancing the state of the art in AKE security models. Our models formally incorporate PCS, and are strictly stronger than (i.e. imply) the best known game-based definitions.
3) We give two specific approaches to achieve PCS. The first allows certain computations with the long-term key but does not reveal it fully. We prove that PCS can be achieved in this case by relatively standard protocols.
4) The second approach permits the adversary to learn the long-term key of an agent before attempting an impersonation attack. We show that no stateless protocol can achieve this type of security, but by providing a generic protocol transformation show that stateful protocols can achieve it. Our transformation produces protocols that are robust to an unreliable network, which turns out to be non-trivial.
5) We supplement our work with case studies showing the practical applicability of post-compromise security. We show with examples of OPTLS and TextSecure/Axolotl that practical protocols can fail to achieve PCS.

Overview. The paper is structured as follows. In Section 2 we discuss the high-level design of security models and how to incorporate PCS. In Section 3 we recall concrete game-based models for AKE protocols, and in Section 4 we extend them formally. We discuss related work in Section 5 and conclude in Section 6. We provide proofs for all our formal statements in the appendices.

2. Modelling PCS

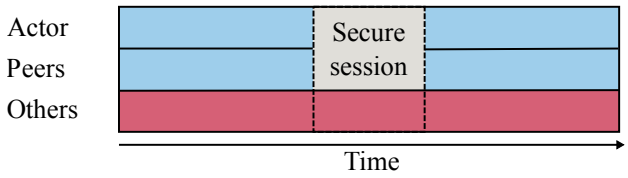
We now turn to the question of constructing a formal model for post-compromise security. In this section we discuss the design choices at a high level, applicable to multiple paradigms within protocol design. In §3, we instantiate our models concretely in the context of two-party AKE protocols.

The execution of a protocol can be abstractly depicted as in Section 2. Each agent (a row in the tables) performs actions over time that may lead to the creation of multiple concurrent sessions. To violate the security guarantee, the adversary must choose a particular agent and session to attack; we call the target agent the 'actor' and the particular session the 'secure' or 'test' session. The secure session communicates with other agents (its peers) while other unrelated actors may also send and receive messages. Our high-level goal is to design protocols that achieve security even against a powerful adversary: we always assume that the network is fully untrusted, and that messages may be modified, dropped or inserted in transit; in addition, we can give the adversary access to actors' secret keys or random number generators. The more powers we grant to the adversary, the 'stronger' the security model and the more attacks the protocol can resist.

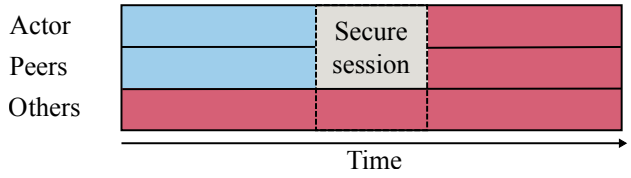
A classical adversary model, depicted in Figure 1(a), is to grant the adversary access to the long-term keys of any party not involved in the secure session. This requires the protocol to limit the scope of compromise: if Charlie's key is revealed, Alice should still be able to communicate securely with Bob. We can strengthen this model by allowing the adversary to compromise all long-term private keys after the end of the attacked session, as depicted in Figure 1(b). This corresponds to (perfect) forward secrecy [11], whereby once a session is completed even the long-term keys of all participants do not suffice to recover the session key. We can also consider an adversary who compromises Alice's private key, as depicted in Figure 1(c). (Recall that we have fixed "Alice" as the actor of the secure session.) This describes a type of attack known as *key compromise impersonation* [1, 19], in which an adversary uses Alice's private key to impersonate as others to Alice.

Figure 1(c) depicts a modern, strong adversary model. It is clear that there is one more step we could take to further strengthen it: to allow compromise in the final region. This corresponds to our concept of post-compromise security, and in this work we will show that it is possible to satisfy security against adversaries which have some powers even in that region. The standard overapproximation is to rule out any compromise at all, as per the figure. To continue the colour analogy, however, Figures 1(d) and 1(e) show two ways in which a form of compromise can still be permitted: we can either colour *nearly* all of the post-compromise box red (dark), or we can colour all of it *nearly* red. We develop both of these ideas, and show that they naturally correspond to different approaches for achieving PCS.

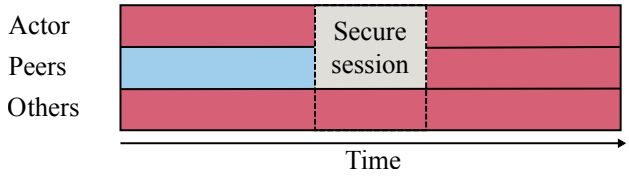
1. Open Whisper Systems designed TextSecure, a secure messaging protocol using a key rotation scheme they called Axolotl. "TextSecure" also refers to their original messaging application, partially incorporated into WhatsApp and now superseded by Signal.



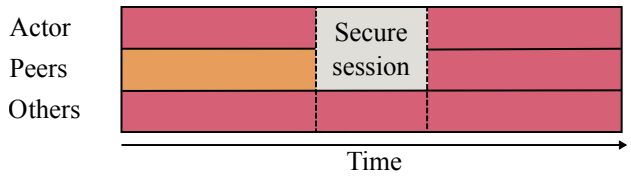
(a) **Classical adversary model.** Abstract representation of the adversary's capabilities during an attack. If Alice is communicating with Bob, classical models allow the adversary to compromise the long-term keys of everyone except Alice and Bob. This ability to compromise the long-term key of non-participants ("others") is depicted in red (dark).



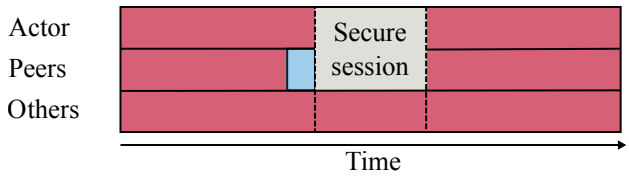
(b) **Adversary model with PFS.** This figure depicts the capabilities of an adversary that can also compromise all long-term private keys after the end of the attacked session. Protocols secure under such a threat model are said to also offer perfect forward secrecy (PFS).



(c) **Adversary model with KCI/AKC.** Later works also consider adversaries that can compromise the actor's long-term private key at any time, and showed that a secure session can still be achieved. In AKE literature, such protocols are said to offer resilience against Key Compromise Impersonation (KCI).



(d) **Adversary model with PCS through limited compromise.** We denote with the orange box a limited form of compromise, which allows the adversary some extra power without revealing the long-term key.



(e) **Adversary model with PCS through state.** Another form of PCS can be achieved if the long-term key is compromised but there is at least one uncompromised session afterwards.

Figure 1: Adversary models for security protocols. Time flows from left to right, and we mark in red (dark) the regions in which we grant the adversary permission to corrupt the agent's secrets. Figure 1(a) represents a relatively weak model, in which the adversary is only allowed to corrupt agents who are not participating in the session under attack. Figures 1(b) and 1(c) extend this model with forward security and actor key compromise resilience. Our stronger adversary models, Figures 1(d) and 1(e), capture post-compromise security.

PCS via weak compromise. Figure 1(d) describes a limited form of compromise permissible against the peers of the secure session. The impersonation attack requires the adversary to use the long-term key value during the secure session. If the model allows some access to the key *without* revealing it to the adversary, and then revokes this access during the secure session, a PCS guarantee may still be achievable.

Claim (informal)**.** *If the adversary is not granted the key itself, but instead limited access to it that is revoked during the secure session, then PCS can still be achieved.*

Practically, this roughly corresponds to a form of trusted key storage, for instance in a hardware security module (HSM). The adversary may have temporary access to the HSM at any point, but without the ability to extract the key. If this access is then revoked, a well-designed protocol may regain its security guarantees.

PCS via state. A more subtle option, as per Figure 1(e), is to colour only some of the box; that is, to allow the long-term key to be revealed before the secure session as long as some uncompromised session has taken place.

The intuition is as follows. Suppose the secure session is between Alice and Bob, and the adversary already knows Bob's long-term key. Suppose further that Alice and Bob have already completed a previous session, deriving some secret value v. If Bob's actions in the secure session depend on v—a value from a previous session—then the adversary cannot necessarily impersonate him. Thus, one session can be used to augment the security of another.

Claim (informal)**.** *If the adversary can compromise all but one exchange of messages before the secure session, then PCS can still be achieved.*

This approach is *stateful*: data from one session must be available to another. (Otherwise, even though Alice and Bob might establish a secret value unknown to the adversary, they could not use that value later for authentication purposes.) We discuss stateful protocols further in Section 4.2.

3. Background on AKE Security Models

In order concretely to instantiate our analysis of PCS in the domain of AKE protocols, we now revisit existing AKE analyses. We refer the reader to e.g. [7, 13, 23] for complete definitions. The majority of security models for authenticated key exchange derive from [3]; recent models of note are the CK model [11] and the eCK model [23], which introduce ephemeral key reveals under various names and can be extended [12] to cover other properties such as forward secrecy. A related model was used for the proof of HMQV [14]. These models have consistently been applied to stateless protocols.

State Reveals. The traditional way to model compromise of ephemeral keys (e.g. x in a Diffie-Hellman key exchange which sends g^x) is to give the adversary a specific power to reveal them. This is natural, but requires designers explicitly to specify which values are considered compromisable. Moreover, the definition of security depends critically on this choice. For example, is x or $H(\mathsf{sk}, x)$ the "ephemeral key" of a NAXOS-style AKE protocol sending $g^{H(\mathsf{sk},x)}$? To avoid this specification problem more recent models instead reveal the outputs of the random number generator, with which any intermediate value can be computed. We follow this trend, and use a randomness query instead of a specific ephemeral key reveal.

Long- and short-term secrets. Security models generally distinguish between long-term secrets (e.g. asymmetric keys used to identify agents) and short-term secrets (e.g. ephemeral keys which are used and subsequently erased). The justification is that practical scenarios exist in which one, but not the other, might be compromised.

There is an ongoing debate about the practicality of this distinction, and in particular whether there are realistic scenarios in which an adversary might compromise an ephemeral key but not its owner's long-term key. We view this debate as interesting but out of scope for the current paper, and follow academic tradition in distinguishing between randomness and corrupt queries.

3.1. Notation and Assumptions

We write $v \leftarrow x$ to denote the assignment of x to the variable v, and $x \leftarrow_\$ X$ to denote that the variable x is assigned a value randomly chosen according to the distribution X. If S is a finite set, we write $x \leftarrow_\$ S$ to mean that x is assigned a value chosen uniformly at random from S. We say that an algorithm is *efficient* if its running time is polynomial in the length of its input, and that a function $f(k)$ is *negligible* if for all $c > 0$ there exists a k_0 such that for all $k > k_0$, $|f(k)| < k^{-c}$. In the context of security protocols, we think of functions and bounds as being negligible in the security parameter(s) of a protocol.

Consider a (multiplicative) cyclic group G of order q with generator g. We make use of the decisional Diffie-Hellman (DDH) hardness assumption, which says that it is hard to distinguish (g^x, g^y, g^{xy}) from (g^x, g^y, g^z), i.e., $\epsilon_{\mathrm{DDH}}(\mathcal{D})$ is negligible in q for any efficient $\mathcal{D}$, where $\epsilon_{\mathrm{DDH}}(\mathcal{D}) \coloneqq \left| \Pr\left[x, y \leftarrow_\$ \mathbb{Z}_q : \mathcal{D}(g^x, g^y, g^{xy}) = 1\right] - \Pr\left[x, y, z \leftarrow_\$ \mathbb{Z}_q : \mathcal{D}(g^x, g^y, g^z) = 1\right] \right|$.

We define a signature scheme $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ to be existentially unforgeable under adaptive chosen message attack (EUF-CMA) via the following game:

Setup. The challenger runs KGen. It gives the adversary the resulting public key pk and keeps the private key sk to itself.

Signature queries. The adversary issues signature queries $m_1, \ldots, m_q$. To each query m_i, the challenger responds by producing the signature σ_i of m_i and sending σ_i to the adversary. These queries may be made adaptively.

Output. Finally, the adversary outputs a pair (m, σ). The adversary wins if σ is a valid signature of m according to Vf and (m, σ) is not among the pairs (m_i, σ_i) generated during the query phase.

If the adversary can only win this game with negligible probability, we say the signature scheme is EUF-CMA.

All hash functions in this paper are assumed to be random oracles. The random oracle model has been extensively studied and simplifies the analysis of hash functions. It is known to admit theoretical attacks [10], though none are yet practical.

3.2. Oracles and Queries

We work in the tradition of Bellare and Rogaway [3], modelling agents (Alice, Bob, ...) as a collection of oracles $\Pi^s_{i,j}$ representing player i talking to intended communication partner j in the s^{th} session. The adversary, Eve, is a Probabilistic Polynomial-Time Turing Machine (PPTM) with access to these "role oracles", who communicate only with Eve and never with each other. As well as relaying messages, Eve has access to certain other powers which are formally implemented as

name	description
s_{actor}	the identity of the user executing session s, where $s_{\text{actor}} \in \mathcal{P}$
s_{role}	the role (initiator $\mathcal{I}$ or responder $\mathcal{R}$) played by s_{actor} in s
s_{peer}	the intended communication partner of s_{actor}, where $s_{\text{peer}} \in \mathcal{P}$
s_{key}	the session key established during the session
s_{rand}	the randomness used in the session
s_{step}	the next step to be executed in the protocol
s_{status}	the session status $s_{\text{status}} \in \{\texttt{active}, \texttt{accepted}, \texttt{rejected}\}$
$s.m_i$	the i^{th} message sent
s_{sent}	all messages sent by s_{actor} during the session
s_{recv}	all messages received by s_{actor} during the session

TABLE 1: Contents of session memory for session s.

query	description
$\mathsf{create}(\hat{A}, r, \hat{B})$	create a new session oracle at $\hat{A}$ with role r and peer $\hat{B}$; randomness is sampled for s_{rand} and Ψ executed to update the state and return a message
$\mathsf{send}(\hat{A}, i, \hat{m})$	execute Ψ on the i^{th} session oracle of $\hat{A}$, update the state, and return the result
$\mathsf{corrupt}(\hat{A})$	reveal the long-term key of $\hat{A}$
$\mathsf{randomness}(\hat{A}, i)$	reveal s_{rand}, where $s = (\hat{A}, i)$
$\mathsf{hsm}(\hat{A}, \dots)$	defined in §4.1
$\mathsf{session\text{-}key}(\hat{A}, i)$	reveal s_{key}, where $s = (\hat{A}, i)$
$\mathsf{cr\text{-}create}(\hat{A}, r, \hat{B}, rnd)$	create a session as with create, but set $s_{\text{rand}} = rnd$ instead of sampling it afresh
$\mathsf{test\text{-}session}(s)$	flip a coin $b \leftarrow\!\$\ \{0,1\}$ and return k_b, where $k_1 = s_{\text{key}}$ and $k_0 \leftarrow\!\$\ \mathsf{KGen}$
$\mathsf{guess}(b')$	end the game

TABLE 2: The set $\mathcal{Q}$ of queries available to $\mathcal{A}$.

"queries". Many different queries have been studied [4, 11, 23] and used to model various security properties. A protocol in this context succeeds if (i) it is correct, in that the parties who intend to talk to each other (and only they) derive matching communication keys, and (ii) it is secure, so that Eve cannot distinguish the test session key from random.

We use notation similar to [13]. Each user can execute any arbitrary number of instances of an AKE protocol π, called sessions. Sessions are uniquely identified by their agent and the order in which they are created. They are implemented by a PPTM whose memory contains both session-specific memory (denoted st_s for each session s) and long-term user memory (denoted $st_{\hat{P}}$ and shared between all sessions).

Session memory st_s is described in Table 1 and is local to a particular session. In contrast, user memory $st_{\hat{P}}$ is shared amongst all sessions of the user $\hat{P}$, and consists of $\hat{P}$'s public/private key pair $(\mathsf{pk}_{\hat{P}}, \mathsf{sk}_{\hat{P}})$, the identities and corresponding public keys of all other users, as well as any other additional information required by the protocol. We assume that the protocol algorithm runs only one step of a session at a time, though many sessions may execute concurrently.

A protocol π is specified by a key generation algorithm KGen and a protocol algorithm Ψ executed by each session oracle. This algorithm takes as input the current session, user state, and an incoming message, and returns an outgoing message as well as new session and user state values.

To define protocol security a game is played in Φ. A setup algorithm first generates user identifiers and keypairs, sets all session-specific variables to an initial value $\perp$, initalises the user memory of each user including distributing all public keys, and initialises all other variables to $\perp$. We choose not to model dynamic key registration for simplicity and because we do not anticipate it leading to any problems unique to the topics addressed in this paper. The adversary then plays the game with access to some subset $Q \subseteq \mathcal{Q}$ of the queries listed in Table 2. Since unrestricted use of these queries can trivially break any session, we will impose some restrictions.

Definition 2 (Matching and partial matching)**.** *Let π be a protocol of $h \geq 2$ total messages, where, if h is even, the number of messages sent from both sides is equal and if h is odd, the initiator sends one extra message. Let s denote a session of π with $s_{status} = \texttt{accept}$. We say that session s* partially matches *session s' in status $s'_{status} \neq \perp$ if the following conditions hold, where $s_{send}[1 \dots l]$ denotes the concatenation of the first l messages sent by session s and $s_{recv}[1 \dots l]$ denotes the concatenation of the first l messages received by s.*

- $s_{role} \neq s'_{role} \wedge s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor}$ *and either*
- $s_{role} = \mathcal{I} \wedge s_{send}[1 \dots m] = s'_{recv}[1 \dots m] \wedge s_{recv}[1 \dots m] = s'_{send}[1 \dots m]$ *with* $m = \frac{h}{2}$ *if h is even and* $m = \frac{h-1}{2}$ *if h is odd, or*
- $s_{role} = \mathcal{R} \wedge s_{send}[1 \dots (m-1)] = s'_{recv}[1 \dots (m-1)] \wedge s_{recv}[1 \dots m] = s'_{send}[1 \dots m]$ *with* $m = \frac{h}{2}$ *if h is even and* $m = \frac{h+1}{2}$ *if h is odd.*

If in addition $(s_{status}, s_{sent}, s_{recv}) = (s'_{status}, s'_{recv}, s'_{sent})$ then we say that s matches *s'.*
If instead $s'_{status} \neq \perp$ and $s'_{send} = s_{recv}$ then we say that s' is an origin-session *[12] for s.*

Definition 3 (Freshness predicate)**.** *A* freshness predicate *is a Boolean predicate which takes a session of some protocol π and a sequence of queries (including arguments and results).*

Definition 4 (AKE security model)**.** *Let π be a protocol in Φ of $h \geq 2$ total messages, let Q denote $\mathcal{A}$'s set of queries, and let F be a freshness predicate. We call (Q, F) an* AKE security model*.*

Definition 5 (Security game)**.** *Let π be a protocol of $h \geq 2$ total messages and $X = (Q, F)$ be an AKE security model. We define the experiment $W(X)$ as follows:*

(i) The setup algorithm is executed.
(ii) The adversary learns all public information (e.g., user identifiers and public keys) and then computes arbitrarily, performing a sequence of queries from the set $Q \setminus \{\mathsf{test\text{-}session}, \mathsf{guess}\}$.
(iii) At some point during $W(X)$, the query $\mathsf{test\text{-}session}(s)$ is issued on some session s that has accepted and satisfies F at the time the query is issued.
(iv) The adversary may continue issuing queries from Q under the condition that s continues to satisfy F.
(v) $\mathcal{A}$ outputs a bit b' via the query $\mathsf{guess}(b')$ as his guess for the bit b chosen during the $\mathsf{test\text{-}session}$ query. The game ends, and $\mathcal{A}$ wins iff $b = b'$.

Following [7, 8], we split the security requirement into two parts: correctness (matching is correctly implemented and implies agreement on the derived key) and secrecy (the adversary cannot distinguish the derived key from random).

Definition 6 (AKE correctness)**.** *A protocol π is said to be* correct *in the security model (Q, F) if, for all PPTM adversaries $\mathcal{A}$, it holds that:*
(i) If completed sessions s and s' match each other, then $s_{key} = s'_{key}$.
(ii) $\Pr[\mathsf{Multiple\text{-}Match}^{\pi,\mathcal{A}}_{W(X)}(k)]$ is negligible, where Multiple-Match *denotes the event that there exists a session admitting two matching sessions.*

Definition 7 (AKE security)**.** *A protocol π is said to be* secure *in the security model (Q, F) if, for all PPTM adversaries $\mathcal{A}$, the $W(X)$-advantage of $\mathcal{A}$, defined below, is negligible in the security parameter.*

$$\mathsf{Adv}^{\pi,\mathcal{A}}_{W(X)}(k) := |\Pr(b = b') - 1/2|$$

Models. We now have sufficient machinery to specify a security model, through a set of queries and a freshness predicate. To demonstrate the concept, we provide an example of a relatively weak model suitable for protocols that are not designed to resist compromised random number generators.

Definition 8 (Basic security model)**.** *The model X_{basic} is defined by $Q = \{\mathsf{create}, \mathsf{send}, \mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}, \mathsf{test\text{-}session}, \mathsf{guess}\}$ and $F = \bigwedge_{k=1}^{5} F_k$ where*
(F_1) No $\mathsf{session\text{-}key}(s)$ query has been issued.
(F_2) For all sessions $s^\star$ such that $s^\star$ matches s, no $\mathsf{session\text{-}key}(s^\star)$ query has been issued.
(F_3) No $\mathsf{randomness}(s)$ query has been issued.
(F_4) For all sessions $s^\star$ such that $s^\star$ partially matches s, no $\mathsf{randomness}(s^\star)$ query has been issued.
(F_5) If there exists no origin-session for session s, then no $\mathsf{corrupt}(s_{peer})$ query has been issued before the create query creating session s or as long as $s_{status} = \mathtt{active}$ and s_{recv} is empty.

It is a standard result that signed Diffie-Hellman (DH) is secure in this type of model under typical cryptographic assumptions.

Theorem 9. *For an EUF-CMA signature scheme $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$, the signed DH protocol of Figure 2 is secure in the model X_{basic} under the DDH assumption.*

Proof: This will follow from Theorem 11, in the next section. ■

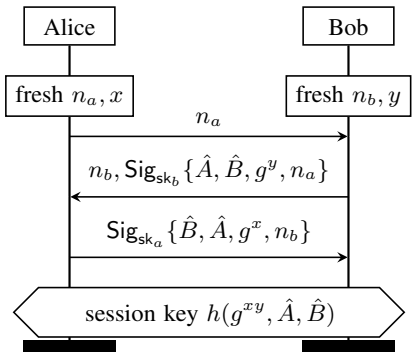


Figure 2: A simple signed challenge-response DH protocol.

4. PCS in AKE

This section will instantiate the informal definitions and claims from Section 2 using the game-based security models described in Section 3. We first consider the case of weak compromise before returning to the full access case.

4.1. Weak Compromise

In order to model limited access to peers' identity keys before the test session, we add a new query $\mathsf{hsm}(\ldots)$ to the model. This query will effectively act as a limited version of corrupt with which the adversary can attempt impersonation attacks.

A model may additionally define a *long-term key interface* $\mathcal{I}$, which is simply a function; we write $(Q, F, \mathcal{I})$ for the model (Q, F) with interface $\mathcal{I}$. We define the query $\mathsf{hsm}(A; \ldots)$ to return $\mathcal{I}(A, \mathsf{sk}_A, \ldots)$, that is, it invokes the long-term key interface with the secret key of its first argument, passing all other arguments through. We use the term "model" both when $\mathcal{I}$ is defined and where it is not (e.g. in other work). This is accomplished by setting $\mathcal{I} = id$ to be the identity function, so that hsm is equivalent to corrupt.

Definition 10. *Let* pcs-hsm *denote the trace predicate that*
(i) for all queries $\mathsf{hsm}(x, \ldots)$, $x = s_{peer}$*; and*
(ii) all queries $\mathsf{hsm}(x, \ldots)$ *were issued before the session* s *was created.*

For any model (Q, F) with $\mathsf{hsm} \notin Q$, we can define a PCS version of that model $(Q \cup \{\mathsf{hsm}\}, F \cap \mathsf{pcs\text{-}hsm})$ that is at least as strong. Informally, this predicate captures that the adversary is allowed to query hsm only on the peer to the test session, and then only before the creation of the test session. This is a strong model: suppose we permitted hsm queries after the creation of the test session. If security were still achievable then the interface is not capturing a meaningful form of compromise, since even with access to it the adversary still cannot impersonate its owner. Moreover, permitting hsm queries against unrelated parties after the completion of the test session does not add any additional adversarial power, since the strictly stronger corrupt query is also permitted at those times.

We can vary the security requirements by choosing $\mathcal{I}$. Naturally, there is an impossibility result for stateless protocols in the case $\mathcal{I} = id$. A protocol is stateless when the inputs to each session are limited to the long-term key and public information, and values computed in one session are not available to others. Here, the hsm query simply reveals the long-term key, so if the protocol carries no state across different sessions, the adversary can use hsm to reveal the long-term key, generate her own session randomness, and then simulate the computations of the peer for a successful impersonation attack.

Is there a meaningful interface $\mathcal{I}$ such that security in a model $(Q \cup \{\mathsf{hsm}\}, F \cap \mathsf{pcs\text{-}hsm}, \mathcal{I})$ is achievable? Yes: a simple challenge-response signed DH protocol provides security in the model where $\mathcal{I}(X, \mathsf{sk}_X, m) = \mathsf{Sig}_{\mathsf{sk}_X}(m)$ is a signature with the long-term key of the target. That is, the adversary is given signatures by Bob on its choice of messages, and still is prevented from impersonating Bob.

Theorem 11. *Let* $\mathcal{I}(A, \mathsf{sk}_A, m) = \mathsf{Sig}_{\mathsf{sk}_A}(m)$ *be the signature algorithm of an EUF-CMA signature scheme* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$*, let* π *be the protocol in Figure 2, and recall* $X_{basic} = (Q, F)$ *from Definition 8. Then* π *is secure in the model* $(Q \cup \{\mathsf{hsm}\}, F \cap \mathsf{pcs\text{-}hsm})$ *under the DDH assumption.*

The proof of Theorem 11 is given in Appendix A. An applied reading of the result is the following: "If Alice's signing key is stored in a hardware security module and used only for the protocol in Figure 2, then to impersonate Alice the adversary must have current access to the HSM." Thus, the protocol achieves PCS under the weak model.

TLS 1.3. Simply storing the long-term key in an HSM is not sufficient for PCS. For example, there has been recent discussion on the new TLS 1.3 specification (the upcoming standard for secure communication over the Internet). One proposal, OPTLS [22], has servers generate and sign temporary public keys that are then used for a signature-free handshake. The intuitive property of the protocol in Figure 2 that grants PCS is that access to the signing key before launching an attack on a session does not help the adversary. In OPTLS, this is not the case: these pre-signed keys effectively act as credentials for the server, and an adversary with temporary access to the signing key can issue fake ones that persist after access is revoked. This was briefly discussed on the TLS mailing list [15, 21], mainly in the context of an attacker who compromises only an ephemeral key. We see that the existence of these credentials prevents OPTLS from satisfying PCS.

4.2. Full Compromise

Theorem 11 achieves a form of PCS when the adversary's access to the long-term key is restricted, allowing it only to query the interface $\mathcal{I}$. However, there are many scenarios in which such a restriction cannot be implemented. Thus, a more interesting case occurs when the adversary has full access to (i.e., *learns*) the long-term key. In this case, we consider $\mathcal{I} = id$, or equivalently allow corrupt queries before the start of the test session.

Recall that we model state compromise through randomness queries: to learn the current state of an agent X, the adversary issues $\mathsf{corrupt}(X)$ as well as randomness queries against all sessions created at X. We are therefore able to model an adversary who learns the entire contents of an agent's memory.

We model this scenario as follows. We begin by defining a minimal model (KE-PCS) which *only* allows the attacker to make PCS attacks, and show that KE-PCS is not satisfiable by stateless protocols. Next, we provide two examples of adding PCS guarantees to existing models: first to a standard eCK-like definition (eCK$^{\mathrm{w}}$, eCK$^{\mathrm{w}}$-PCS), and second to a strong model (Ω_{AKE}) for the class of all AKE protocols. Finally, we show that our models are achievable.

In order to resist PCS attacks, a protocol may negotiate a per-peer intermediate secret (IS) each time it completes a session, deriving it deterministically from the session's randomness and messages. The IS is used in future sessions to derive the session key as well as a successor IS. We shall call such a design a *ratcheting protocol*[2], following the terminology of Axolotl [26]: intuitively, the IS ratchets forward each time a session is completed.

Once a ratcheting protocol has "got off the ground" so to speak, it maintains secrecy of the IS until future compromise. Moreover, if any IS is secret then all of its successors are also secret unless specifically compromised. We formalise this secret-establishment phase as follows. Recall that cr-create represents creation of a session with chosen randomness, and is therefore a more powerful version of the randomness query.

Definition 12 (Ratcheted session)**.** *A session s is* ratcheted *if there exists an "intermediate" session s' with matching session s'' such that*

(i) $s_{actor} = s'_{actor} \wedge s_{peer} = s'_{peer}$,
(ii) *s' and s'' both accept before the creation of s,*
(iii) *at most one of* $\mathsf{corrupt}(s'_{actor})$ *and* ($\mathsf{randomness}(s')$ *or* $\mathsf{cr\text{-}create}(s')$ *creating session s') have been issued, and*
(iv) *at most one of* $\mathsf{corrupt}(s'_{peer})$ *and* ($\mathsf{randomness}(s'')$ *or* $\mathsf{cr\text{-}create}(s'')$ *creating session s'') have been issued.*

Definition 13 (KE-PCS)**.** *The model* KE-PCS *is defined by (Q, F) where $Q = \{\mathsf{create}, \mathsf{send}, \mathsf{corrupt}\}$ and $F = P_1 \wedge P_2$:*
(P_1) the corrupt *query occurs, if issued, against s_{peer} and before the creation of s*
(P_2) if there is no origin-session for s, and the corrupt *query was issued, then s is ratcheted*

The model KE-PCS captures precisely (and only) the concept of full PCS, the blue area in Figure 1(e): if the adversary compromises the long term keys of the peer before the test session starts, and the test session is not ratcheted, then it can impersonate the peer. (In this case there is no origin-session for the messages received by the test session.)

The ratcheting construction is relatively complex. In particular, it requires sessions to be able to share state with each other: the IS value derived in one session is an input to the computations in the next. We show now that state is necessary for security in KE-PCS.

Theorem 14. *No stateless protocol is secure in* KE-PCS.

The KE-PCS model captures one particular aspect of protocol security. In order to give a comprehensive model, we can extend existing AKE models to include PCS guarantees. For example, we recall and extend the eCK$^{\mathrm{w}}$ model [12]. This yields a practical model for key exchange protocols which captures eCK security as well as PCS.

Definition 15 (eCK$^{\mathrm{w}}$, eCK$^{\mathrm{w}}$-PCS)**.** *Consider the clauses defined in Definition 8 and Table 3. The models eCK^{w} and eCK^{w}-PCS are defined by their freshness predicates as follows.*

$$F(\textit{eCK}^{w}) = F_1 \wedge F_2 \wedge F_3^{\textit{eCK}^{w}} \wedge F_4^{\textit{eCK}^{w}} \wedge F_5^{\textit{eCK}^{w}}$$
$$F(\textit{eCK}^{w}\textit{-PCS}) = \quad \cdots \quad \wedge F_5^{\textit{KR}}$$

Remark 16. *The models eCK [23] and eCK^{w} differ only in the case that the adversary wishes to attack a session s admitting an origin-session $s^\star$ which is not a matching session for s. We use eCK^{w} as our example since its presentation is closer to our own, but see no reason that the same arguments should not apply also to eCK.*

There are stronger models than eCK$^{\mathrm{w}}$, and we may naturally attempt to capture as many attacks as we can. Cremers and Feltz [13] define strong models for classes of protocols (Appendix B); in the spirit of their work, by including PCS attacks into their strongest model we can construct a strong model for the class of all AKE protocols. We follow their naming scheme, whereby the model Ω_{X} is a strong model for the class X of protocols.

Definition 17 ($\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$)**.** *Consider the clauses defined in Definition 8 and Table 3. The models $\Omega_{AKE\cap ISM}$ [13] and Ω_{AKE} are defined by their freshness predicates as follows.*

$$F(\Omega_{\textit{AKE}\cap\textit{ISM}}) = F_1 \wedge F_2 \wedge F_3^{\textit{ISM}} \wedge F_4^{\textit{ISM}} \wedge F_5^{\textit{SL}}$$
$$F(\Omega_{\textit{AKE}}) = \quad \cdots \quad \wedge F_5^{\textit{KR}}$$

2. Some of the documentation on ratcheting protocols says that they "ratchet the long-term key". This is equivalent to our formulation of keeping a fixed long-term key and storing a separate IS.

Name	Definition
F_1	No $\mathsf{session\text{-}key}(s)$ query has been issued.
F_2	For all sessions $s^\star$ such that $s^\star$ matches s, no $\mathsf{session\text{-}key}(s^\star)$ query has been issued.
$F_3^{\mathsf{eCK^w}}$	Not both queries $\mathsf{corrupt}(s_{\text{actor}})$ and ($\mathsf{randomness}(s)$ or $\mathsf{cr\text{-}create}(.)$ creating session s) have been issued.
$F_4^{\mathsf{eCK^w}}$	For all sessions s' such that s' is an origin-session for s, not both queries $\mathsf{corrupt}(s_{\text{peer}})$ and ($\mathsf{randomness}(s')$ or $\mathsf{cr\text{-}create}(.)$ creating session s') have been issued.
$F_5^{\mathsf{eCK^w}}$	If there exists no origin-session for session s, then no $\mathsf{corrupt}(s)$ query has been issued.
F_3^{ISM}	Not all queries $\mathsf{corrupt}(s_{\text{actor}})$ as well as ($\mathsf{randomness}$ or $\mathsf{cr\text{-}create}$) queries on all $\hat{s}$ with $\hat{s}_{\text{actor}} = s_{\text{actor}}$, where the query ($\mathsf{create}$ or $\mathsf{cr\text{-}create}$) creating session $\hat{s}$ occurred before or at the creation of session s, have been issued.
F_4^{ISM}	For all sessions s' where s partially matches s', not all queries $\mathsf{corrupt}(s_{\text{peer}})$ as well as ($\mathsf{randomness}$ or $\mathsf{cr\text{-}create}$) queries on all sessions $\hat{s}$ with $\hat{s}_{\text{actor}} = s'_{\text{actor}}$, where the query ($\mathsf{create}$ or $\mathsf{cr\text{-}create}$) creating session $\hat{s}$ occurred before or at the creation of session s', have been issued.
F_5^{KR}	If there exists no origin-session for session s, and a $\mathsf{corrupt}(s_{\text{peer}})$ query has been issued before creation of session s via a query (create or $\mathsf{cr\text{-}create}$) or as long as $s_{\text{status}} = \mathtt{active}$ and $s_{\text{recv}} = \epsilon$, then s is ratcheted.

TABLE 3: Clauses of the freshness predicates for our AKE models. The predicates with superscript eCK^w and ISM come from [12] and [13] respectively. The new predicate is the final one, marked with superscript KR.

Rationale for the freshness conditions. The Ω models described in Appendix B are constructed as follows. First, impossibility results are proven for a class of protocols, by constructing adversaries which provably succeed against all protocols in the class. Second, these adversaries are ruled out through careful choice of a freshness predicate. For example, the predicate $F_3^{\text{eCK}^\text{w}}$ rules out the class of adversaries which compromise both the randomness of the test session and the long-term key of its actor, since such adversaries are guaranteed always to succeed against stateless protocols. We take these strong models as our base models, thereby inheriting these freshness predicates.

The new predicate, F_5^{KR}, encodes the PCS attacks captured by KE-PCS: if there is no origin-session for s (i.e. the adversary is impersonating its agent), and the key of s_{peer} has been revealed, then there must have been a "blue area" or ratcheting session allowing security to be re-established.

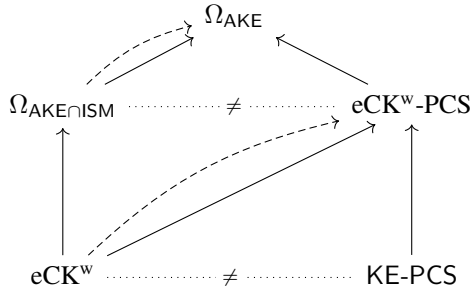


Figure 3: Relationship between the various models we define in this section. Solid arrows denote inequality of security models ($\leq_{sec}$, Appendix B), dotted arrows denote incomparability (neither model implies the other), and dashed arrows denote our transformation. For example, in this set Ω_{AKE} provides the strongest security guarantees.

The P_1 and P_2 predicates defined in KE-PCS are much more restrictive than the F_j predicates we use subsequently, ruling out many attacks which a protocol could resist. Specifically, they limit the adversary to learning the key *only* of the peer to the test session. This is by design; KE-PCS is not intended as a strong model.

Satisfiability of the Models. Defining these models is of no use if they are not satisfiable: for a model X to be interesting we must be able to prove that some nontrivial protocol is secure in X.

In Section 4.1 we defined a concrete protocol and proved it secure. Here, the construction required to ensure that state is synchronised between communicating parties, as well as correctly used in the key derivation, is significantly more complex. Instead of giving a concrete protocol, therefore, we define a generic transformation $\pi \mapsto \pi^\dagger$.

This transformation takes a three-message protocol π and converts it into a ratcheting protocol that achieves PCS (and possibly more). Specifically, if π is secure in $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ then we prove that $\pi^\dagger$ is secure in Ω_{AKE}.

The use of state for authentication has a number of subtleties, and to deal with them our final transformation is relatively complex. In particular, if Alice updates her state without explicitly authenticating her peer, she risks deleting a secret value that she needs to communicate with Bob. Thus, stateful implicitly authenticated protocols with PCS can have a severe denial-of-service flaw. More precisely, we define the notion of *robustness* as follows.

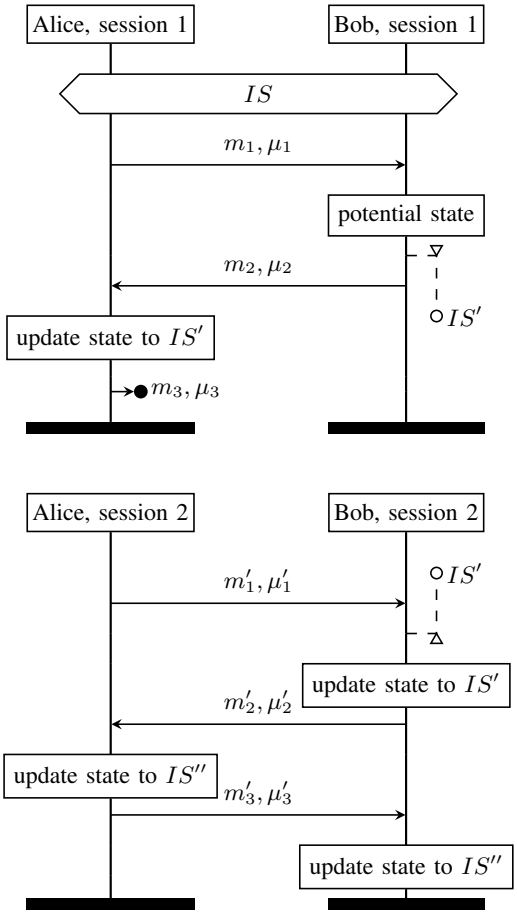
Definition 18 (Robustness)**.** *Let $\mathcal{C}(\hat{A}, \hat{B})$ be the benign "correct execution" adversary who creates an initiator session at $\hat{A}$, relays its initial message to $\hat{B}$, returns the response to $\hat{A}$ and continues until $\hat{A}$ completes (including relaying the final message to $\hat{B}$). A* network adversary *is any adversary who only issues* create *and* send *queries.*

A protocol π is robust *if for any $\hat{A}$ and $\hat{B}$, $\mathcal{C}$ causes a pair of matching sessions deriving the same key to be established at $\hat{A}$ with $\hat{B}$ and vice versa. It is* post-network robust *if $\mathcal{C}(\hat{A}, \hat{B})$ still induces a pair of matching sessions which derive the same session key when executed sequentially after any network adversary.*

Let π be a three-message protocol $\hat{A} \xrightarrow{m_1} \hat{B} \xrightarrow{m_2} \hat{A} \xrightarrow{m_3} \hat{B}$. Assume that public DH keys are distributed for all participants, and that π computes a pre-master secret from which session keys are derived. The initial intermediate secret $IS_0^{\hat{P}_i,\hat{P}_j}$ for any two parties $\hat{A}, \hat{B}$ is set to g^{ab}

- New global state variables $IS^{\hat{B},\hat{A}}$ and $st_{\hat{B}}^{\mathcal{R}}.\text{potential}_{\hat{A}}$ are defined at each agent $\hat{B}$ for each other agent $\hat{A}$, the first storing a single value and the second a possibly empty list of values.
- In a session s between parties $\hat{A}$ and $\hat{B}$, each message $\hat{A} \xrightarrow{m_j} \hat{B}$ is replaced by $\langle m_j, \mu_j \rangle$ where $\mu_j = \mathsf{MAC}(m_j; IS^{\hat{A},\hat{B}})$, and each message $\hat{B} \xrightarrow{m'_j} \hat{A}$ is replaced by $\langle m'_j, \mu'_j \rangle$ where $\mu'_j = \mathsf{MAC}(m'_j; IS^{\hat{B},\hat{A}})$.
- Upon receiving $\langle m_1, \mu_1 \rangle$, $\hat{B}$ acts as follows:
  - If $\mu_1 = \mathsf{MAC}(m_1; IS^{\hat{B},\hat{A}})$, $\hat{B}$ continues to the next step.
  - If not, $\hat{B}$ checks for each value $is \in st_{\hat{B}}^{\mathcal{R}}.\text{potential}_{\hat{A}}$ whether $\mu_1 = \mathsf{MAC}(m_1; is)$. If this holds for some value is, $\hat{B}$ replaces $IS^{\hat{B},\hat{A}} \leftarrow is$, empties $st_{\hat{B}}^{\mathcal{R}}.\text{potential}_{\hat{A}} \leftarrow \emptyset$, and continues to the next step.
  - Otherwise, $\hat{B}$ `rejects`.
- The new value $IS_{\text{new}}^{\hat{B},\hat{A}} \leftarrow \text{KDF}(\sigma, IS^{\hat{B},\hat{A}}, 1)$ is appended to $st_{\hat{B}}^{\mathcal{R}}.\text{potential}_{\hat{A}}$.
- The pre-master secret $pms \leftarrow \text{KDF}(\sigma)$ is replaced by $pms' \leftarrow \text{KDF}(\sigma, IS^{\hat{B},\hat{A}}, 0)$.
- $\hat{B}$ computes $\xrightarrow{m_2}$ and sends $\langle m_2, \mu_2 \rangle$
- $\hat{A}$ verifies μ_2 using its intermediate secret, and `rejects` if the verification fails. Otherwise, it updates $IS^{\hat{A},\hat{B}} \leftarrow IS_{\text{new}}^{\hat{A},\hat{B}} = \text{KDF}(\sigma, IS^{\hat{A},\hat{B}}, 1)$, and sends $\langle m_3, \mu_3 \rangle$.
- $\hat{B}$ verifies μ_3 against the potential value from the current session, and if the verification passes and it would `accept` then it first sets $IS^{\hat{B},\hat{A}} \leftarrow IS_{\text{new}}^{\hat{B},\hat{A}}$ and $st_{\hat{B}}^{\mathcal{R}}.\text{potential}_{\hat{A}} \leftarrow \emptyset$.

(a) Definition of the PCS transform $\pi \mapsto \pi^{\dagger}$.



(b) Example trace of a transformed protocol whose original messages were m_1, m_2, m_3. We show two sessions between Alice and Bob: in the first, the final message is dropped so Bob does not update to IS'. In the second, Bob recalls IS' from earlier (represented by the dotted line) and performs the earlier missed update.

Figure 4: Two-round, robust PCS transform.

Theorem 19. *No correct one-round protocol which is secure in* KE-PCS *is post-network robust.*

This theorem implies that *any* one-round protocol that achieves security in KE-PCS is vulnerable to a denial of service in which an adversary with temporary network control can prevent any two parties from ever communicating again. This is in stark contrast to the world of stateless protocols, where if the first session run by $\mathcal{C}$ accepts with some probability, then any subsequent session also accepts with the same probability.

We give the final version of the protocol transform, which handles these and other issues, in Figure 4(a). In Figure 4(b) we show an example execution that illustrates the robustness mechanism. We refer the reader to the appendices for a full explanation of the issues and justification of the transform design. The intuition is that it adds explicit authentication (through a MAC applied to each message and requiring knowledge of the current IS), in such a way that Alice only updates an intermediate secret if she can be confident that Bob can derive the new value. The modified protocol $\pi^{\dagger}$ is closely related to π^{*}, and we argue for security of the former based on that of the latter. (In fact, protocols transformed from eCK$^{\text{w}}$ are secure in a stronger model than eCK$^{\text{w}}$-PCS: intermediate secret acts as a weak entropy pool, sharing randomness across sessions. This weak pooling is not very practical, however, since a stronger version is achievable with minimal overhead. The formal details are included in the appendices.)

Theorem 20. *Let π be a stateless protocol.*

(i) If π is secure in eCK^w then $\pi^\dagger$ is secure in eCK^w-PCS.
(ii) If π is secure in $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$, then $\pi^\dagger$ is secure in Ω_{AKE}.
(iii) If π is robust then $\pi^\dagger$ is post-network robust.

The proof proceeds in several stages and is included in the appendices. We first give a simpler transformation $\pi \mapsto \pi^*$, which replaces the session key derivation $K \leftarrow \mathrm{KDF}(pms)$ with $K \leftarrow \mathrm{KDF}(0, pms, IS)$ and generates a new $IS \leftarrow \mathrm{KDF}(1, pms, IS)$. Assuming it is hard to invert the KDF, we show that an adversary who can compute the IS output by a session can also compute the session key of that session. Next, we show that due to the "chaining" of IS values, it is hard for the adversary to derive the IS of a ratcheted session. From these observations we can prove security of the simpler transformation in e.g. Ω_{AKE}.

Next, we consider an intermediate protocol π^*_{MAC}, which adds the MAC values μ_j to the messages but does not maintain the buffer of potential new state values. We show that the addition of the MACs does not weaken the security of the protocol, and therefore that π^*_{MAC} is also secure in Ω_{AKE}. Finally, to complete the security proof we show that the transformed protocol $\pi^\dagger$ is at least as secure as π^*_{MAC}, by reducing an adversary against the former to one against the latter.

Results in the computational setting are generally negative, in the sense that they prove that the adversary cannot distinguish between certain values. Robustness is not a property of this form, and therefore the standard game-hopping techniques are not immediately applicable. We instead construct a security argument to prove item (iii), showing that certain syntactic invariants imply that parties must remain synchronised. Roughly, we prove an invariant that if one party updates the state $s \mapsto s'$ then the other party "knows" s', in the sense that s' is either its current state or present in its buffer of potential updates. By construction of the protocol algorithm, this suffices to show that the correct-execution adversary $\mathcal{C}$ indeed induces matching sessions.

5. Related Work

There is relatively little work on related phenomena. The concept of "key continuity" appears in various Internet Drafts [9, 17], but there is little formal work in the academic literature to mirror these discussions.

Various protocols appear to implement some sort of key continuity (or PCS) features, including OTR [16], ZRTP [9], and even many SSH [27] implementations via the known-hosts feature. However, existing analyses of these protocols do not cover this aspect. Bowers et al. [6] define a notion of "drifting keys", in which devices randomly modify their long-term key over time, such that a trusted verifier can detect the discrepancies caused by an attacker with a slightly-incorrect key. This type of protocol detects the adversary but does not prevent the compromise. Nevertheless, we view their work as providing a type of PCS. Itkis [18] describes forward secrecy for primitives as part of a more general theory of "key evolution", in which keys are constantly rotated. This gives a type of PCS, though applying not to protocols but to the cryptographic primitives themselves.

Stateful protocols. Cremers and Feltz [13] consider protocols that use shared state to pool randomness between different sessions, but do not consider synchronisation of state between agents. Our work builds on theirs.

There has not been much analysis of stateful AKE, but state is used in various other primitives: stateful symmetric encryption is well-known, and asymmetric encryption is also discussed in the literature [2]. State is of course used *within* sessions—for example, ACCE [20] provides security to the message channel in the sense of stateful length-hiding authenticated encryption.

Future secrecy. The author of [25] describes a property of the Axolotl ratchet called "future secrecy", defined as resistance to an adversary which is *passive* after compromising the key of a peer. We remark that such adversaries are covered by the notion of (weak perfect) forward secrecy: their passivity means they do not need to use the compromised data until *after* choosing the test session, so they could achieve the same result by delaying the compromise until after the choice of test session. Thus, post-compromise secrecy with respect to a passive adversary is already covered by forward secrecy. Indeed, such security can be achieved by a basic, unsigned Diffie-Hellman key exchange, meaning that it is more relevant in the context of lightweight or otherwise restricted classes of protocols.

The (recently discontinued) TextSecure messaging app, a major user of Axolotl, did not in fact admit PCS due to its implementation of multi-device messaging: it shared long-term keys across all devices but maintained separate state, so that an adversary could impersonate Bob by claiming to be a previously-unseen device. As a consequence, it is clear if the ratcheting mechanism in TextSecure provided any additional *security* guarantees above and beyond those of a strong AKE protocol, though it certainly provided message transport with fewer round-trips.

6. Conclusions

In hindsight, post-compromise security seems a highly desirable feature of communication systems, and it is perhaps surprising that it has not been formalised before. Devices are frequently compromised, and post-compromise security is an achievable property in a scenario that has not been previously considered. Our technical contributions in the domain of AKE protocols demonstrate what might be achieved by post-compromise secure protocols, and how this can be achieved.

We showed that the use of a hardware security model to store keys can allow the effects of a key compromise to be undone, so that an adversary can no longer carry out impersonation attacks. There are relatively simple protocols which achieve this level of security, although not all protocols do.

We then considered the more general case that the adversary learns the secret key, and defined a model in which the adversary can do so as long as an unattacked session is permitted to complete. We observed that in this scenario stateless protocols cannot prevent impersonation. However, some stateful protocols can do so: we gave a generic transformation producing stateful protocols, and proved that its outputs are secure in our models. Not all ratcheting protocols necessarily achieve PCS, however, and we showed that indeed the protocol in TextSecure did not.

As the notion of post-compromise security applies more widely than to just AKE protocols, we expect related areas also to pick up on this concept and harden their constructions. This requires additional formal work, but should proceed along the lines sketched here. We remark in particular that our first model is directly relevant to users who want the benefits of storing keys in HSMs.

Some more direct avenues of future work include:

1) We considered only the case of a signature HSM; that is, a key storage interface which produces only signature with the long-term key. Of course, most HSMs perform a wide range of operations. A natural extension of our work would consider a more general interface to the key, inspired perhaps by the PKCS#11 standard.
   Such an extension comprises a 'balancing act' between applicability and ease of proof. For example, Diffie-Hellman-style protocols often require key derivation operations of the form $(-)^{\mathsf{sk}}$. The interaction between such an interface and the Diffie-Hellman computations of the session key is subtle.
   In the spirit of strong models, we also foresee an interesting avenue of research investigating the limits of HSM-style queries. For example, is there a natural class of interfaces for which no PCS is possible?
2) We considered a concrete protocol and proved it secure in the presence of a signature interface. A natural extension of this question is to search for necessary or sufficient conditions for arbitrary protocols to meet this goal. For example, it is clear that the signature should contain some fresh data.
3) A feature of Axolotl [26] which we did not discuss is that it sends messages with zero round trips. An interesting question is the extent to which the ratcheting mechanism is necessary to achieve this, and how the security goals for a 0-RTT protocol interact with PCS.

References

[1] D. Basin, C. Cremers, and M. Horvat. "Actor Key Compromise: Consequences and Countermeasures". In: Computer Security Foundations Symposium (CSF). July 2014, pp. 244–258. DOI: 10.1109/CSF.2014.25.

[2] Mihir Bellare, Tadayoshi Kohno, and Victor Shoup. "Stateful Public-key Cryptosystems: How to Encrypt with One 160-bit Exponentiation". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. New York, NY, USA: ACM, 2006, pp. 380–389. ISBN: 978-1-59593-518-2. URL: http://doi.acm.org/10.1145/1180405.1180452 (visited on 11/10/2015).

[3] Mihir Bellare and Phillip Rogaway. "Entity Authentication and Key Distribution". In: *Advances in Cryptology – CRYPTO '93*. Ed. by Douglas R. Stinson. Lecture Notes in Computer Science 773. Springer Berlin Heidelberg, Jan. 1, 1994, pp. 232–249. ISBN: 978-3-540-57766-9 978-3-540-48329-8. URL: http://link.springer.com/chapter/10.1007/3-540-48329-2_21 (visited on 05/21/2014).

[4] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. "Key Agreement Protocols and Their Security Analysis". In: *Proceedings of the 6th IMA International Conference on Cryptography and Coding*. London, UK, UK: Springer-Verlag, 1997, pp. 30–45. ISBN: 978-3-540-63927-5.

[5] Nikita Borisov, Ian Goldberg, and Eric Brewer. "Off-the-record Communication, or, Why Not to Use PGP". In: *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*. WPES '04. New York, NY, USA: ACM, 2004, pp. 77–84. ISBN: 1-58113-968-3. URL: http://doi.acm.org/10.1145/1029179.1029200 (visited on 06/22/2015).

[6] K.D. Bowers et al. "Drifting Keys: Impersonation detection for constrained devices". In: *2013 Proceedings IEEE INFOCOM*. 2013 Proceedings IEEE INFOCOM. Apr. 2013, pp. 1025–1033. DOI: 10.1109/INFCOM.2013.6566892.

[7] Christina Brzuska et al. "Composability of Bellare-rogaway Key Exchange Protocols". In: CCS. ACM, 2011, pp. 51–62. ISBN: 978-1-4503-0948-6. URL: http://doi.acm.org/10.1145/2046707.2046716 (visited on 07/27/2015).

[8] Christina Brzuska et al. *Less is More: Relaxed yet Composable Security Notions for Key Exchange*. 242. 2012. URL: http://eprint.iacr.org/2012/242 (visited on 08/18/2015).

[9] Jon Callas, Alan Johnston, and Philip Zimmermann. *ZRTP: Media Path Key Agreement for Unicast Secure RTP*. Apr. 2011. URL: https://tools.ietf.org/html/rfc6189#page-107 (visited on 02/10/2016).

[10] Ran Canetti, Oded Goldreich, and Shai Halevi. *The Random Oracle Methodology, Revisited*. 011. 1998. URL: http://eprint.iacr.org/1998/011 (visited on 04/08/2014).

[11] Ran Canetti and Hugo Krawczyk. “Analysis of key-exchange protocols and their use for building secure channels”. In: Springer-Verlag, 2001, pp. 453–474.

[12] Cas Cremers and Michèle Feltz. “Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal”. English. In: *Designs, Codes and Cryptography* (2013), pp. 1–36. ISSN: 0925-1022. URL: http://dx.doi.org/10.1007/s10623-013-9852-1.

[13] Cas Cremers and Michèle Feltz. *On the Limits of Authenticated Key Exchange Security with an Application to Bad Randomness*. https://eprint.iacr.org/2014/369.pdf. 2014.

[14] Cas J. F. Cremers. *Formally and Practically Relating the CK, CK-HMQV, and eCK Security Models for Authenticated Key Exchange*. 253. 2009. URL: http://eprint.iacr.org/2009/253 (visited on 07/27/2015).

[15] Daniel Kahn Gillmor. *[TLS] OPTLS: Signature-less TLS 1.3*. Nov. 2, 2014. URL: https://www.ietf.org/mail-archive/web/tls/current/msg14423.html (visited on 02/11/2016).

[16] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. “Secure Off-the-record Messaging”. In: *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*. WPES ’05. New York, NY, USA: ACM, 2005, pp. 81–89. ISBN: 1-59593-228-3. (Visited on 01/30/2014).

[17] Peter Gutmann. *Key Management through Key Continuity (KCM)*. 2008. URL: https://tools.ietf.org/html/draft-gutmann-keycont-01 (visited on 02/10/2016).

[18] Gene Itkis. “Forward Security: Adaptive Cryptography – Time Evolution”. In: *Handbook of Information Security*. Vol. 3. John Wiley and Sons, 2006.

[19] Mike Just and Serge Vaudenay. “Authenticated Multi-Party Key Agreement”. In: *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology*. ASIACRYPT ’96. London, UK, UK: Springer-Verlag, 1996, pp. 36–49. ISBN: 978-3-540-61872-0. URL: http://dl.acm.org/citation.cfm?id=647093.716554 (visited on 02/09/2016).

[20] Florian Kohlar, Sven Schäge, and Jörg Schwenk. *On the Security of TLS-DH and TLS-RSA in the Standard Model*. 367. 2013. URL: https://eprint.iacr.org/2013/367 (visited on 07/27/2015).

[21] Hugo Krawczyk. *[TLS] OPTLS: Signature-less TLS 1.3*. Nov. 12, 2014. URL: https://www.ietf.org/mail-archive/web/tls/current/msg14592.html (visited on 02/11/2016).

[22] Hugo Krawczyk and Hoeteck Wee. *The OPTLS Protocol and TLS 1.3*. 978. 2015. URL: https://eprint.iacr.org/2015/978 (visited on 02/01/2016).

[23] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. *Stronger Security of Authenticated Key Exchange*. 073. 2006. URL: http://eprint.iacr.org/2006/073 (visited on 02/26/2014).

[24] Adam Langley. *Pond*. 2014. URL: https://pond.imperialviolet.org/ (visited on 06/22/2015).

[25] Moxie Marlinspike. *Advanced cryptographic ratcheting*. Nov. 26, 2013. URL: https://whispersystems.org/blog/advanced-ratcheting/ (visited on 02/09/2016).

[26] Trevor Perrin. *trevp/axolotl*. GitHub. 2014. URL: https://github.com/trevp/axolotl (visited on 06/22/2015).

[27] Tatu Ylonen and Chris Lonvick. *The Secure Shell (SSH) Transport Layer Protocol*. 2006. URL: https://tools.ietf.org/html/rfc4253 (visited on 02/10/2016).

Appendix A.
Proofs for the Restricted-Compromise Model

Theorem 11. *Let $\mathcal{I}(A, \mathsf{sk}_A, m) = \mathsf{Sig}_{\mathsf{sk}_A}(m)$ be the signature algorithm of an EUF-CMA signature scheme $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$, let π be the protocol in Figure 2, and recall $X_{basic} = (Q, F)$ from Definition 8. Then π is secure in the model $(Q \cup \{\mathsf{hsm}\}, F \cap \mathsf{pcs\text{-}hsm})$ under the DDH assumption.*

Proof sketch : Denote the event of the adversary winning in Game i by S_i. Assume the adversary always selects a test session s where $s_{\mathsf{actor}} = \hat{A}$ is the initiator. The other case is similar. For the adversary to win, the test session must have accepted. Let the transcript of messages sent and received be as in Figure 2.

Game 0: This is the original security game.

Game 1: This is the original security game except it aborts if two nonces ever collide or if two ephemeral DH shares ever collide. This is a standard gamehop, and we see that $|\Pr(S_0) - \Pr(S_1)|$ is negligible.

Game 2: This game is identical to Game 1 except it aborts if the challenger fails to correctly guess the test session s before the game begins. This is another standard game hop.

Game 3: This game is identical to game 2 except it aborts if the adversary ever queries $\mathsf{hsm}(\hat{B}, \hat{A}, g^y, n_a)$ (for the unique g^y received by $\hat{A}$ in the second message of s and the unique nonce n_a selected by $\hat{A}$ in the initial message of s). As access to the hsm query is revoked before n_a is even generated by $\hat{A}$, the probability that the adversary will query hsm on the random string n_a is negligible. Hence $|\Pr(S_2) - \Pr(S_3)|$ is negligible.

Game 4: This game is identical to Game 3 except it aborts if the received signature $\mathsf{Sig}_{\mathsf{sk}_b}\{\hat{A}, \hat{B}, g^y, n_a\}$ for actor $\hat{A}$ in session s is forged, where $s_{\mathsf{peer}} = \hat{B}$. Since the signature scheme is EUF-CMA, previous signatures, either from other protocol runs or the hsm query, will not give the adversary non-negligible advantage in making this forgery. Hence $|\Pr(S_3) - \Pr(S_4)|$ is negligible.

By Game 3 and Game 2, we must have that the signature is genuinely created by $\hat{B}$ in a (or several) session(s), since it cannot be forged or found via the hsm query. Also, since the ephemeral DH value g^y is unique by Game 1, the session s' the signature was created in is unique.

Game 5: This game is identical to Game 4 except it aborts if the challenger does not successfully guess s' such that $s_{\mathsf{peer}} = \hat{A}$ and $\hat{B}$ selected g^y in session s' to send to $\hat{A}$ in session s. This is a standard game hop.

Game 6: This game is identical to Game 5 except we replace g^{xy} as computed in the test session key of s and (possibly s' if the last message is forwarded) by g^z. This is a standard game hop based on the decisional Diffie-Hellman assumption. The hop is allowed since the only two sessions that could compute g^{xy} cannot be queried by the adversary.

As the session key contains the hash of g^z for random z, the session key is indistinguishable from random. Thus,the adversary cannot possibly have any non-negligible advantage in winning this game. By our hops, the protocol is secure in the original security game.

■

Appendix B.
Strong Models for Protocol Classes

We here recall some definitions from Cremers and Feltz [13], including the concept of a strong model for a protocol class.

Definition 21. *Let $secure(M, \pi)$ be a predicate that is true if and only if the protocol π is secure in security model M, and let Π be a class of AKE protocols. We say that a security model M' is* at least as strong *as a security model M with respect to Π, denoted by $M \leq^{\Pi}_{Sec} M'$, if*

$$\forall \pi \in \Pi, secure(M', \pi) \implies secure(M, \pi)$$

We say that M and M' are incomparable *if $M \not\leq^{\Pi}_{Sec} M' \wedge M' \not\leq^{\Pi}_{Sec} M$.*

Definition 22 (Protocol classes)**.** *We define AKE as the class of all protocols of $h \geq 2$ total messages for which the probability that two sessions of the same user output in all protocol steps identical messages is negligible.*

The subclass ISM of AKE (for "initial state modification") consists of all protocols in AKE that only access and update user memory upon creation of sessions.

The subclass SL of ISM consists of all stateless protocols; that is, those for which $st_{\hat{p}} = proj_3(\Psi(1^k, st_s, st_{\hat{p}}, m))$ for all $(k, st_s, st_{\hat{P}}, m)$, where $proj_3$ is projection onto the third coordinate.

Each of these classes admits different strong models: attacks which can be prevented in a larger class may always work in a smaller one. For example, we have already seen the impossibility result that stateless protocols cannot achieve key indistinguishability if the long-term key of the peer to the test session has been revealed.

[13, Theorem 2] states five generic attacks against protocols in $\mathsf{AKE} \cap \mathsf{SL}$, and rules them out in the security model below (there denoted $\Omega^{-}_{\mathsf{AKE}}$). The first pair of attacks are clear: the adversary directly queries the session key of the test session or

its partner. The second pair correspond to an adversary which corrupts all secret inputs to a party and thereby simulates its computation. In the final attack the adversary corrupts some party and then impersonates them to the test session.

Definition 23 ($\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$)**.** *The $\Omega_{AKE\cap SL}$ model is defined by $(\mathscr{Q}, F)$ where $F = F_1 \wedge F_2 \wedge F_3^{SL} \wedge F_4^{SL} \wedge F_5^{SL}$ and the F_j are as in Table 3 and as follows:*

(F_3^{SL}) Not both queries corrupt(s_{actor}) *and (*randomness(s) *or* cr-create(.) *creating session s) have been issued.*

(F_4^{SL}) For all sessions s' such that s is partially matching s', not both queries corrupt(s_{peer}) *and (*randomness(s') *or* cr-create(.) *creating session s') have been issued.*

(F_5^{SL}) If there exists no origin-session for session s, then no corrupt(s_{peer}) *query has been issued before creation of session s via a (*create *or* cr-create*) query or as long as $s_{status} =$* `active` *and $s_{recv} = \epsilon$.*

A similar impossibility result, analogous to [13, Corollary 3], applies to protocols in ISM. Since in this class randomness is shared between parties, it suffices for at least one previous session to have uncorrupted randomness.

Definition 24 ($\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$)**.** *The model $\Omega_{AKE\cap ISM}$ is defined by $(\mathscr{Q}, F)$ where $F = F_1 \wedge F_2 \wedge F_3^{ISM} \wedge F_4^{ISM} \wedge F_5^{SL}$ and the clauses are as per Table 3.*

We remark that the translation from an impossibility result to a strong model is not unique. An impossibility result gives specific adversaries which can attack any protocol in a class, and a corresponding strong model rules out those adversaries but as few others as possible. In particular, if an attack is not as generic as thought, the corresponding freshness predicate will rule out more adversaries than strictly necessary—for example, the "generic" PCS adversary only attacks the first session, but the final clause of the freshness predicate rules out any compromise before the test session.

We now define some models for use in subsequent proofs.

Definition 25. *We define the following freshness predicates.*

(F_3^{KR}) If both queries corrupt(s_{actor}) *and (*randomness(s) *or* cr-create *creating session s) have been issued, then s is ratcheted. item For all sessions s' such that s is partially matching s', if both queries* corrupt(s_{peer}) *and (*randomness(s') *or* cr-create *creating session s') have been issued, then s' is ratcheted.*

(F_4^{KR}) If there exists no origin-session for session s, and a corrupt(s_{peer}) *query has been issued before creation of session s via a query (*create *or* cr-create*) or as long as $s_{status} =$* `active` *and $s_{recv} = \epsilon$, then s is ratcheted.*

We define the following models by their freshness predicates.

$$F(\mathcal{S}_{\min}) = F_1 \wedge F_2 \wedge F_3^{SL} \wedge F_4^{SL} \wedge F_5^{KR}$$
$$F(\mathcal{S}_{SL}) = F_1 \wedge F_2 \wedge F_3^{KR} \wedge F_4^{KR} \wedge F_5^{KR}$$

The difference between $F(\mathcal{S}_{\min})$ and $F(\text{eCK}^{\text{w}}\text{-PCS})$ is only in F_4: the former restricts compromise only on sessions $s^\star$ which *partially match* the test session s, while the latter restricts compromise on any $s^\star$ which is an *origin-session* for s. Since a partially matching session for s is an origin session for s, we see that $F(\text{eCK}^{\text{w}}\text{-PCS}) \implies F(\mathcal{S}_{\min})$. Likewise, it is clear that $F(\mathcal{S}_{\min}) \implies F(\mathcal{S}_{\mathsf{SL}})$, since the latter again allows strictly more adversarial actions. Hence:

Lemma 26. *It holds that eCK^w-PCS $\leq_{sec} \mathcal{S}_{\min} \leq_{sec} \mathcal{S}_{SL}$.*

B.1. Pooling of Randomness

The key difference between the above models is in the treatment of the third and fourth clauses of the freshness predicate, which describe the security of sessions where both an agent's long-term key and local randomness are compromised. Specifically, this situation is forbidden in SL since there they comprise the only secret inputs to the session key computation, but there is more than one natural way to lift the restriction outside of SL.

A protocol which stores the randomness of all past sessions (or some derived value e.g. a hash or seeded PRNG) needs only one of these to be uncorrupted to maintain session key secrecy. This is the intuition behind $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$. However, because it only stores local secrets, the state does not help authenticate its peer.

A protocol which stores state synchronised with its peer can use that state for improved authentication. Because the state depends on previous random values, it can also provide some guarantees if the local randomness for a session is compromised. These guarantees are weaker in two ways, though:

(i) because the secret is per-peer, if the randomness of a session with Bob is compromised then only a previous session *with Bob* can help, and

(ii) because the peer also knows the secret, it could also be leaked *by them*.

Concretely, let $\mathcal{A}$ denote the adversary which honestly relays three sessions: s_1 : Alice $\leftrightarrow$ Bob, s_2 : Alice $\leftrightarrow$ Charlie and s_3 : Alice $\leftrightarrow$ Bob, then queries corrupt(Alice), randomness(s_1) and randomness(s_3). This adversary is valid in $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ since there was at least one previous session with uncorrupted randomness, but invalid in $\mathcal{S}_{\mathsf{SL}}$ since that session was not with Bob. Conversely, the generic impersonation attack is valid in $\mathcal{S}_{\mathsf{SL}}$ but succeeds against any protocol in ISM.

This demonstrates that the models $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ and $\mathcal{S}_{\mathsf{SL}}$ are incomparable. But once state is shared between sessions, there is no obvious reason not to pool randomness as well as maintain a shared secret, giving stronger guarantees if the local RNG is weak. Such protocols' guarantees are given by Ω_{AKE}.

Appendix C. Robustness

Theorem 14. *No stateless protocol is secure in* KE-PCS.

Proof: The only secret inputs to the protocol algorithm of a stateless protocol are the session randomness and the long-term key. It follows that the simple corrupt-and-impersonate adversary has the same success probability as the corrupted agent. ■

One-round protocols with post-compromise secrecy are necessarily not robust to an unreliable network. The intuitive reason is that if the initiator sends a message m but receives no reply, she cannot tell whether

(i) m was lost before delivery, or

(ii) m was received by Bob but the response lost.

For a one-round protocol, the latter case has Bob accept a session with an intermediate secret that Alice does not know, which clearly will cause trouble with future sessions.

We now formalise this argument. By a *benign* adversary we mean one which acts as a wire, faithfully relaying every message in order to its intended peer.

Theorem 19. *No correct one-round protocol which is secure in* KE-PCS *is post-network robust.*

Proof: We give an explicit adversary in KE-PCS against Π. The adversary corrupts some agent Bob, then completes an honest session between Alice and Bob, meaning that Bob's state may change. The adversary then uses the *previous* state value, which it learnt from the corruption, to impersonate Bob to Alice in a subsequent session. We must argue (i) that this adversary does not contradict freshness, which holds because of the intermediate session; and also (ii) that it wins the security game, which we show with an explicit reduction to robustness.

Pick two targets $\hat{A}$ and $\hat{B}$. Since Π is post-network robust, the correct-execution adversary $\mathcal{C} = \mathcal{C}(\hat{B}, \hat{A})$ induces matching sessions at $\hat{A}$ and $\hat{B}$, with the final protocol message sent $\hat{A} \to \hat{B}$. (Otherwise use $\mathcal{C}(\hat{A}, \hat{B})$. Our adversary $\mathcal{A}$ begins by executing $\mathcal{C}$; that is, it simulates its execution, and whenever $\mathcal{C}$ issues a query $\mathcal{A}$ issues the same query, and returns the response to $\mathcal{C}$.

Next, $\mathcal{A}$ issues a $\mathsf{corrupt}(\hat{B})$ query, and uses the response to set up a simulated $\hat{B}$-oracle Σ, initialising it with the setup algorithm used by the game. If $\hat{B}$ is the initiator then it simulates a $\mathsf{create}(\Sigma, \hat{A})$ query; if the responder then it additionally simulates a $\mathsf{send}(\Sigma, m_1, \hat{A})$ query where m_1 was the initial message sent by $\mathcal{C}$ above. (This ensures that the state of Σ is equidistributed with that of $\hat{B}$, required for Case 2 below.)

Now, $\mathcal{A}$ uses Σ to impersonate $\hat{B}$ to $\hat{A}$: it executes $\mathcal{C}$ for a second time, except redirecting all $\mathsf{send}(\hat{B}, \ldots)$ queries made by $\mathcal{C}$ to Σ. That is, $\mathcal{A}$ stores the current state st_Σ in its memory, and whenever $\mathcal{C}$ issues a $\mathsf{send}(\hat{B}, \ldots)$ query $\mathcal{A}$ computes $\Psi(\mathrm{st}_\Sigma, \ldots)$ and returns its output as the result of the query. All other send queries are faithfully invoked by $\mathcal{A}$.

Let s be the session created at $\hat{A}$ by this execution. Note that s must complete, and therefore has status $s.\mathrm{status} \in \{\mathtt{accept}, \mathtt{reject}\}$. We consider these cases separately.

Case 1: s accepts. We argue that s is fresh, and that it derives a key matching the key in st_Σ.

To show that s is fresh there are two conditions in the freshness predicate. The first is that the only queries were issued against $\hat{B}$ before the ratcheting session; this is true. The second is that s is ratcheted; that is, that there exist two matching sessions at $\hat{A}$ and $\hat{B}$, accepted before the creation of s, for which not both $\mathsf{corrupt}$ and ($\mathsf{randomness}$ or $\mathsf{cr\text{-}create}$) queries have been issued. This is indeed true: the two sessions created by $\mathcal{C}$ satisfy this condition. Hence s is fresh.

$\mathcal{A}$ may therefore issue $\mathsf{test\text{-}session}(s)$, and compare the result to the session key in the memory of the unique session s' at the simulated $\hat{B}$-oracle. Since the protocol is correct and s matches s', we must have that $s.\mathrm{key} = s'.\mathrm{key}$ with high probability, and hence the adversary can easily win. In this case the theorem is proved.

Case 2: s rejects. We argue that the marginal distribution at $\hat{A}$ could also be induced by a network adversary.

Consider the network adversary $\mathcal{C}'$ which is equal to $\mathcal{C}$ except that the final message $\hat{A} \to \hat{B}$ is dropped. Consider the distributions of messages sent to $\hat{A}$ by $(\mathcal{C}'; \mathcal{C})$ and by $\mathcal{A}$. The first message is sampled from the distribution of initial messages at $\hat{B}$, possibly with the additional but equal first message. Subsequent messages from the first session at $\hat{B}$ are sampled from equal distributions, since $\Pi \in \mathsf{IFSM}$ implies that $\mathrm{st}_{\hat{B}}$ is not updated by intermediate messages. The first message from $\mathcal{C}$ (resp. Σ) is sampled from still the same distribution, since the final message was not delivered and hence the state at $\hat{B}$ was not updated. Subsequent messages from $\mathcal{C}$ (resp. Σ) are sampled from equal distributions, again because $\Pi \in \mathsf{IFSM}$ implies that $\mathrm{st}_{\hat{B}}$ is not updated by intermediate messages. By this equality of distributions, we have that the session induced by $\mathcal{C}$ in $(\mathcal{C}'; \mathcal{C})$ also $\mathtt{rejects}$ and therefore $\mathcal{C}$ did not create a pair of matching sessions. Since $\mathcal{C}'$ is a network adversary, Π is not post-network robust. In this case the theorem is proved. ■

Sequential precomposition with any adversary has no effect on the output distribution of role oracles in a stateless protocol. We therefore have

Let π be a protocol whose session keys are derived as $KDF(pms)$. Let s denote the i^{th} completed session that has had actor $\hat{A}$ and peer $\hat{B}$, and let $IS_i^{\hat{A},\hat{B}}$ denote the i^{th} intermediate secret $\hat{A}$ has stored for $\hat{B}$ in $st_{\hat{A}}$. The protocol π is modified as follows:

- The session key $k \leftarrow \text{KDF}(\sigma)$ that $\hat{A}$ computes in session s is replaced by $k \leftarrow \text{KDF}(\sigma, IS_{i-1}^{\hat{A},\hat{B}}, 0)$.
- The next intermediate secret for peer $\hat{B}$ that $\hat{A}$ computes is $IS_i^{\hat{A},\hat{B}} \leftarrow \text{KDF}(\sigma, IS_{i-1}^{\hat{A},\hat{B}}, 1)$.
- Upon session acceptance, $IS_i^{\hat{A},\hat{B}}$ is stored in $st_{\hat{A}}$, replacing $IS_{i-1}^{\hat{A},\hat{B}}$ which is erased.

Actor $\hat{B}$ computes intermediate secrets and session keys identically, and the initial intermediate secret $IS_0^{\hat{P}_i,\hat{P}_j}$ for any two parties $\hat{P}_i, \hat{P}_j$ is publicly set to 0.

Figure 5: Definition of the one-round PCS transform $\pi \mapsto \pi^*$

Corollary 27. *Any robust stateless protocol is post-Adv robust, but there exist robust protocols in ISM which are not even post-benign robust.*

Appendix D. PCS Transform

D.1. One-round PCS transform

In this section, we present a generic transform to turn a protocol π in the class ISM into a protocol $\pi^\star$ not in ISM. We will show that $\pi^\star$ achieves security in the models above, but is not robust.

First, we require that π derive its session key from a pre-master secret. This may be assumed without loss of generality, however: adding an extra key derivations step will not reduce security. For the remainder of this section, let $X^\star = (Q^\star, F)$ denote $\mathcal{S}_{\text{min}}$, $\mathcal{S}_{\mathsf{SL}}$ or Ω_{AKE}.

Lemma 28. *Let $\pi \in$ AKE be a protocol that does not use a hash function KDF to compute the session key k as $k \leftarrow \text{KDF}(pms)$ for some pms. Let $\widetilde{\pi}$ be the transformed protocol of π that does. If π is secure in model $X^\star$, then so is $\widetilde{\pi}$.*

Proof: We prove the contrapositive. That is, we prove that if $\widetilde{\pi}$ is insecure in model $X^\star$, then so is π. We do this by constructing an adversary $\mathcal{A}_\pi$ that succeeds against protocol π with non-negligible probability by using a corresponding adversary $\mathcal{A}_{\widetilde{\pi}}$ that succeeds against $\widetilde{\pi}$ with non-negligible probability. Such an adversary as $\mathcal{A}_{\widetilde{\pi}}$ exists because of the assumption that $\widetilde{\pi}$ is insecure.

We construct the adversary $\mathcal{A}_\pi$ against protocol π as follows. $\mathcal{A}_\pi$ runs as a challenger, implementing users executing protocol $\widetilde{\pi}$ against adversary $\mathcal{A}_{\widetilde{\pi}}$. When $\mathcal{A}_{\widetilde{\pi}}$ makes a create, cr-create, send, randomness or corrupt query, $\mathcal{A}_\pi$ asks the same query as an adversary against π and forwards the reply back to $\mathcal{A}_{\widetilde{\pi}}$. When $\mathcal{A}_{\widetilde{\pi}}$ makes a session-key query, $\mathcal{A}_\pi$ asks the same query as an adversary against π, receives reply pms and then forwards $\text{KDF}(pms)$ back to $\mathcal{A}_{\widetilde{\pi}}$. When $\mathcal{A}_{\widetilde{\pi}}$ asks a test-session query, $\mathcal{A}_\pi$ asks the same query as an adversary against π, receives reply k_b and forwards $\text{KDF}(k_b)$ back to $\mathcal{A}_{\widetilde{\pi}}$. Finally, $\mathcal{A}_\pi$ outputs the same guess b' as $\mathcal{A}_{\widetilde{\pi}}$. Since $\mathcal{A}_{\widetilde{\pi}}$ is a successful adversary, the test session for $\mathcal{A}_\pi$ is fresh. Moreover, $\mathcal{A}_\pi$ makes the correct guess for b when $\mathcal{A}_{\widetilde{\pi}}$ does. As $\mathcal{A}_{\widetilde{\pi}}$ succeeds with non-negligible probability, so does $\mathcal{A}$. Therefore, if $\widetilde{\pi}$ is insecure, then so is π. ■

We define a new protocol $\pi^\star$ with the same messages as π but with a different key derivation in Figure 5. If $\hat{A}$ and $\hat{B}$ are synchronised—that is, they have the same intermediate secret for each other—then they will derive equal session keys from an honest session.

We show first that $\pi^\star$ is at least as secure as π, and second that $\pi^\star$ additionally meets security in the PCS models.

Lemma 29. *If π is secure in model $X^\star$, then so is $\pi^\star$.*

Proof: We prove the contrapositive. That is, we prove that if $\pi^\star$ is insecure in model $X^\star$, then so is π. For an adversary $\mathcal{A}_{\pi^\star}$ in model $X^\star$ that succeeds against protocol $\pi^\star$ with non-negligible probability, consider adversary $\mathcal{A}_\pi$, also in model $X^\star$, but against protocol π. $\mathcal{A}_\pi$ proceeds in an almost identical way to $\mathcal{A}_{\pi^\star}$ except for minor differences. $\mathcal{A}_\pi$ ignores intermediate secrets, e.g., where $\mathcal{A}_{\pi^\star}$ computes $\text{KDF}(\sigma, IS, 0)$, $\mathcal{A}_\pi$ instead computes $\text{KDF}(\sigma)$. Where $\mathcal{A}_{\pi^\star}$ uses a session-key(s) query to reveal key $k^\star = \text{KDF}(\sigma, IS, 0)$ which $\mathcal{A}_{\pi^\star}$ then uses in later computations, $\mathcal{A}_\pi$ uses the same session-key(s) query to reveal a different session key $k = \text{KDF}(\sigma)$ which $\mathcal{A}_\pi$ uses instead (subject to the condition of $\mathcal{A}_\pi$ ignoring intermediate secrets as already specified). Subject to these conditions, $\mathcal{A}_\pi$ executes the same trace as $\mathcal{A}_{\pi^\star}$. Denote the test session for $\mathcal{A}_{\pi^\star}$ (resp. $\mathcal{A}_\pi$) against protocol $\pi^\star$ (resp. π) by $s_{\pi^\star}$ (resp. s_π). Similarly let the test session key $k_{\pi^\star}$ (resp. k_π) be computed as $\text{KDF}(\sigma_{\pi^\star}, IS, 0)$ (resp. $\text{KDF}(\sigma_\pi)$). As $\mathcal{A}_{\pi^\star}$ is a successful adversary, $s_{\pi^\star}$ is fresh. By the construction of $\mathcal{A}_\pi$ and the freshness conditions in $X^\star$, s_π is also fresh.

As session keys are not used to compute any other terms in the protocol, and as intermediate secrets are only used to compute other intermediate secrets or session keys, for all intents and purposes $\mathcal{A}_\pi$ implements the same attack as $\mathcal{A}_{\pi^\star}$. If $\mathcal{A}_{\pi^\star}$ wins against $\pi^\star$ by querying $\text{KDF}(\sigma_{\pi^\star}, IS, 0)$, then $\mathcal{A}_{\pi^\star}$ must have deduced the input tuple $(\sigma_{\pi^\star}, IS, 0)$, and in particular $\sigma_{\pi^\star}$, which

is a function of only the key exchange. But since $\mathcal{A}_\pi$ has the same trace as $\mathcal{A}_{\pi^\star}$, $\mathcal{A}_\pi$ is able to deduce σ_π and k_π. Therefore, if the adversary $\mathcal{A}_{\pi^\star}$ succeeds against $\pi^\star$ with this strategy, then $\mathcal{A}_\pi$ can succeed against π. If on the other hand $\mathcal{A}_{\pi^\star}$ wins against $\pi^\star$ while never querying $\mathrm{KDF}(\sigma_{\pi^\star}, IS, 0)$, then as the KDF is a random oracle, the only possibility is that this is a key replication attack. Clearly if $\mathcal{A}_{\pi^\star}$ can do this, then so can $\mathcal{A}_\pi$. ■

Before we prove the main theorem of this section (Theorem 32), we make the following observation. Suppose an adversary has to derive an intermediate secret $IS_i = \mathrm{KDF}(\sigma, IS_{i-1}, 1)$. As the KDF is a random oracle, the only way of doing so is by evaluating the KDF on the input $(\sigma, IS_{i-1}, 1)$. If the adversary can do this, then she can also compute the session key that comes with the same session as IS_i, as $k = \mathrm{KDF}(\sigma, IS_{i-1}, 0)$. If the adversary can compute a session key, then she does not need to use a session-key query on the session (or any matching session). Note we do not have circular reasoning here as session keys are never used to compute anything in the protocol, including intermediate secrets. This leads us to the following two lemmata which we will use in addition to Lemma 29 to prove Theorem 32.

Lemma 30 is cumbersome to write, but states intuitively that computing an intermediate secret from a session with uncompromised state is as difficult as computing the session key of that session.

Lemma 30. *Take either (i) $\Omega = eCK^w$ and $KR = eCK^w$-PCS or (ii) $\Omega = \Omega_{AKE \cap ISM}$ and $KR = \Omega_{AKE}$. Consider all adversaries $\mathcal{A}_{\pi^\star}$ in KR against protocol $\pi^\star$ such that the test session s of $\mathcal{A}_{\pi^\star}$ is ratcheted and consider sessions s' and s'' as in the definition of s being ratcheted. Denote the intermediate secret computed by s_{actor} (and possibly s_{peer} if synchronised) in s' (and possibly s'') by IS_i. So in other words, $IS_i = \mathrm{KDF}(\sigma_{i-1}, IS_{i-1}, 1)$ for some key exchange information σ_{i-1} and previous intermediate secret IS_{i-1}. If protocol π is secure in Ω, then the probability of adversary $\mathcal{A}_{\pi^\star}$ computing the intermediate secret IS_i is negligible.*

Proof: Assume for contradiction that an adversary $\mathcal{A}_{\pi^\star}$ exists that can compute IS_i with non-negligible probability. Recall the observation that if $\mathcal{A}_{\pi^\star}$ can compute IS_i, then $\mathcal{A}_{\pi^\star}$ can also compute the session key of the same session s'. Therefore we can assume without loss of generality that $\mathcal{A}_{\pi^\star}$ does not issue a session-key(s') (or, if synchronised, a session-key(s'') query).

Consider adversary $\mathcal{A}_\pi$ against (secure) protocol π in Ω that copies $\mathcal{A}_{\pi^\star}$ apart from in two ways: (i) $\mathcal{A}_\pi$ ignores intermediate secrets in the same way as in the proof of Lemma 29 and (ii) $\mathcal{A}_\pi$ makes the same queries as adversary $\mathcal{A}_{\pi^\star}$ except where $\mathcal{A}_{\pi^\star}$ queries test-session(s), $\mathcal{A}_{\pi^\star}$ instead queries test-session(s'), and $\mathcal{A}_\pi$ does not necessarily make the same guess query as $\mathcal{A}_{\pi^\star}$. We see that since $\mathcal{A}_{\pi^\star}$ always makes a test session ratcheted, it follows that s' in fresh. Therefore if $\mathcal{A}_{\pi^\star}$ can compute IS_i with non-negligible probability, then $\mathcal{A}_\pi$ can compute the test session key and succeed against π in Ω with non-negligible probability. But π is secure in Ω, which is a contradiction. ■

Lemma 31 generalises this result to a *sequence* of intermediate secrets, since they are chained together inside random oracle invocations. It states intuitively that if an intermediate secret comes from a ratcheted session, then the adversary is unable to compute any subsequent intermediate secret derived from it.

Lemma 31. *Take either (i) $\Omega = eCK^w$ and $KR = eCK^w$-PCS or (ii) $\Omega = \Omega_{AKE \cap ISM}$ and $KR = \Omega_{AKE}$. Consider all adversaries $\mathcal{A}_{\pi^\star}$ in KR against protocol $\pi^\star$ such that the test session s of $\mathcal{A}_{\pi^\star}$ is ratcheted and consider sessions s' and s'' as in the definition of s being ratcheted. Denote the intermediate secrets s_{actor} stores for s_{peer} as $IS_0, IS_1, IS_2 \ldots$ where the intermediate secret computed in session s' is IS_i and the intermediate secret used in the computation of the test session key of session s is IS_j (so $j \geq i$). If π is secure in Ω, then the probability of adversary $\mathcal{A}_{\pi^\star}$ computing the intermediate secret IS_j is negligible.*

Proof: We see that $IS_j = \mathrm{KDF}(\sigma_{j-1}, IS_{j-1}, 1)$ for some key exchange information σ_{j-1}. Intermediate secrets are never sent or received, so for $\mathcal{A}_{\pi^\star}$ to compute IS_j, $\mathcal{A}_{\pi^\star}$ must compute the input IS_{j-1} (as well as σ_{j-1}). But $IS_{j-1} = \mathrm{KDF}(\sigma_{j-2}, IS_{j-2}, 1)$ for some key exchange information σ_{j-2}, and so on. By induction we see that for $\mathcal{A}_{\pi^\star}$ to compute IS_j, $\mathcal{A}_{\pi^\star}$ must compute IS_i. But by Lemma 30 this can only be done with negligible probability. ■

Theorem 32. *For any protocol $\pi \in AKE \cap ISM$,*

(i) If π is secure in eCK^w, then $\pi^\star$ is secure in eCK^w-PCS

(ii) If π is secure in $\Omega_{AKE \cap ISM}$, then $\pi^\star$ is secure in Ω_{AKE}

The high-level proof structure is as follows. For both (ii) and (i) we have already shown that $\pi^\star$ is at least as secure as π by Lemma 29, so all that remains is to cover the cases of additional adversarial behaviour allowed against $\pi^\star$ but disallowed against π. In these additional cases, we argue as in Lemma 30 and Lemma 31 that a ratcheted session protects the necessary intermediate secret and thus the test session key.

Proof: If $\mathcal{A}$ creates a fresh test session we say it is *valid*.

(ii): Since the Ω_{AKE} model is identical to the $\Omega_{\mathsf{AKE} \cap \mathsf{ISM}}$ model except for the difference in the 5th predicate, it follows by Lemma 29 that $\pi^\star$ will be secure in Ω_{AKE} if we can show that for all adversaries $\mathcal{A}$ that are valid in Ω_{AKE} but not in $\Omega_{\mathsf{AKE} \cap \mathsf{ISM}}$, $\mathcal{A}$ can only defeat $\pi^\star$ with negligible probability. In other words, such an adversary $\mathcal{A}$ must create a test session s for which no origin session exists, and also issue a corrupt(s_{peer}) query. We can rule out the case that $\mathcal{A}$ wins by doing this as well as never

computing the test session key $k = \mathrm{KDF}(\sigma_j, IS_j, 0)$ by the same arguments as before. Now, by the 5th predicate in Ω_{AKE}, the test session s must be ratcheted, so there exists two other sessions s' and s'' as in the definition of s being ratcheted. To compute the KDF on the inputs $(\sigma_j, IS_j, 0)$ required for the test session key, the adversary needs the intermediate secret IS_j, but this cannot be computed with non-negligible probability by Lemma 31.
(i): Consider adversary $\mathcal{A}$ in $\mathcal{S}_{\mathsf{SL}}$ against $\pi^\star$. Denote the test session by s with test session key $k = \mathrm{KDF}(\sigma_j, IS_j, 0)$. We partition into the events (i) V, (ii) $V^c \cap T_0^c$, (iii) $V^c \cap T_0 \cap S$ and (iv) $V^c \cap T_0 \cap S^c$; where V, T_0 and S are defined by:

V	$\mathcal{A}$ is a valid adversary in $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$.
T_0	There exists an origin session for the test session.
S	$\mathcal{A}$ issues a $\mathsf{corrupt}(s_{\text{actor}})$ and a ($\mathsf{randomness}(s)$ or $\mathsf{cr\text{-}create}(s)$) query.

Event V. This case is covered by Lemma 29.

Event $V^c \cap T_0 \cap S^c$. If $\mathcal{A}$ is a valid adversary in Ω_{AKE} but not in $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, then it must be because of the difference in the 3rd, 4th or 5th freshness predicates as the 1st and 2nd are identical. If an origin session also exists, it must be because of the 3rd or 4th predicates. If $\mathcal{A}$ also does not issue a $\mathsf{corrupt}(s_{\text{actor}})$ and a $\mathsf{randomness}(s)$ (or $\mathsf{cr\text{-}create}(s)$) query, then the difference must be in the 4th predicate. In this case, since $\mathcal{A}$ is still a valid adversary in Ω_{AKE}, it must be the case that there exists a session s' such that s is partially matching s', where the queries $\mathsf{corrupt}(s_{\text{peer}})$ as well as ($\mathsf{randomness}(s')$ or $\mathsf{cr\text{-}create}(s')$) have been issued, where s' is ratcheted. By analogous reasoning to Lemma 30 and Lemma 31, the intermediate secret IS_j used in the computation of the test session key k is safe, so we are done.

Event $V^c \cap T_0 \cap S$. If $\mathcal{A}$ is a valid adversary in Ω_{AKE} but not in $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, then it must be because of the difference in the 3rd, 4th or 5th freshness predicates as the 1st and 2nd are identical. If an origin session also exists, it must be because of the 3rd or 4th predicates. If it is because of the 4th, see the analysis of event $V^c \cap T_0 \cap S^c$. If it is (possibly also) because of the difference in the 3rd predicate, then by the 3rd predicate of $\mathcal{S}_{\mathsf{SL}}$, s must be ratcheted. By analogous reasoning to Lemma 30 and Lemma 31, the intermediate secret IS_j used in the computation of the test session key k is safe, so we are done.

Event $V^c \cap T_0^c$. If $\mathcal{A}$ is a valid adversary in Ω_{AKE} but not in $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, then it must be because of the difference in the 3rd, 4th or 5th freshness predicates as the 1st and 2nd are identical. If it is because of the difference between the 3rd and 4th predicates, see the analysis of Event $V^c \cap T_0 \cap S$. If it is (possibly also) because of the difference in the 5th predicate, then this case is covered by analogous reasoning to (ii). ■

A useful corollary of Theorem 32 is that one can use it to reverse engineer proofs of security; real-world protocols that use synchronised state may be complex and lack proofs of security, but if they are of the same form as $\pi^\star$ then it will suffice to prove the security of the simpler corresponding protocol of the same form as π.

D.2. Two-Round PCS Transform

The previous PCS transform was not robust when applied to multiple-round protocols. We give an alternative transform, specific to two-round protocols, which does achieve robustness. Again, it converts a protocol π subject to some restrictions into a protocol $\pi^\dagger$ not in ISM.

Intuitively, the major obstacle to robustness is related to the two generals' problem: if Alice updates her state, the next message might be dropped. However, since we postcondition on the fact that $\mathcal{C}$'s messages are completed unaltered, we solve this by allowing the first message to play the role of the dropped one. We therefore build a protocol transform $(\cdot)^\dagger$ which is closely related to $(\cdot)^\star$, but (i) the initial messages are authenticated with the current state, (ii) secrets are tied to the particular protocol role of each party, and (iii) there is logic to handle the case where a final message was dropped. Specifically, each party maintains separate initiator and responder states; the latter contains not only the current IS value but also a collection of potential ones, corresponding to sessions which might have been accepted by initiator $\hat{A}$. Upon receipt of the final protocol message, the responder $\hat{B}$ updates its state and deletes all other potential values. However, in case this final message was dropped, it also checks incoming initial messages against the current potential states and updates if necessary.

The formal description of $\pi^\dagger$ was depicted in Figure 4(a), and its message flows in Figure 4(b). It assumes that public DH keys are distributed for all parties, and that π as before computes a pre-master secret from which session keys are derived.

Theorem 20. *Let π be a stateless protocol.*

(i) If π is secure in eCK^w then $\pi^\dagger$ is secure in eCK^w-PCS.
(ii) If π is secure in $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$, then $\pi^\dagger$ is secure in Ω_{AKE}.
(iii) If π is robust then $\pi^\dagger$ is post-network robust.

The high-level argument for Theorem 20 is as follows. We alter the protocol $\pi^\star$ in minor steps until it becomes protocol $\pi^\dagger$, arguing about the security of each protocol along the way. First, we transform π into $\pi^\star$ to gain the claimed extra security by Theorem 32. We then consider an intermediate protocol $\pi^\star_{\mathsf{MAC}}$ and show it is at least as secure as $\pi^\star$. Finally, we show

that $\pi^\dagger$ is at least as secure as $\pi^\star_{\mathsf{MAC}}$. At the end, we will have shown that $\pi^\dagger$ is at least as secure as $\pi^\star_{\mathsf{MAC}}$, which is at least as secure as $\pi^\star$, which is secure in the required model by Theorem 32.

Proof sketch: We only prove that security in $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ implies security in Ω_{AKE}; the other implication is analogous.

First transform π into $\pi^\star$ using the PCS transform of Section D. By Theorem 32, $\pi^\star$ is secure in Ω_{AKE}.

Next, transform $\pi^\star$ into $\pi^\star_{\mathsf{MAC}}$, which is identical to $\pi^\star$ except each message m_j is replaced with $\langle m_j, \mu_j\rangle$ as in the transform above. Note that $\pi^\star_{\mathsf{MAC}}$ is not robust as it does not use a global $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ variable as in the transform above. However, $\pi^\star_{\mathsf{MAC}}$ is also secure in Ω_{AKE}; if an adversary can succeed against $\pi^\star_{\mathsf{MAC}}$ in Ω_{AKE} with non-negligible probability, then essentially the same attack works on $\pi^\star$. If the attack requires the test session being fresh and ratcheted, then we can apply the random oracle assumption and use the hash inside a HMAC to show that no extra information is gained from $\pi^\star_{\mathsf{MAC}}$ over $\pi^\star$. Hence by Lemma 31 the attack must not involve the adversary gaining any knowledge of the IS, and as such the same attack (but ignoring the MACs) can work against $\pi^\star$. If on the other hand the attack works with the test session not being ratcheted, then no security of the IS is guaranteed and so the MAC is superfluous information. As such, an attack of this type on $\pi^\star_{\mathsf{MAC}}$ implies an attack on $\pi^\star$. So any attack on $\pi^\star_{\mathsf{MAC}}$ can be translated onto an attack on $\pi^\star$. Thus $\pi^\star_{\mathsf{MAC}}$ is secure in Ω_{AKE}.

To prove $\pi^\dagger$ is at least as secure as $\pi^\star_{\mathsf{MAC}}$, we assume the existence of a successful adversary $\mathcal{A}_{\pi^\dagger}$ against $\pi^\dagger$ and use it to construct a successful adversary $\mathcal{A}_{\pi^\star_{\mathsf{MAC}}}$ against $\pi^\star_{\mathsf{MAC}}$. For the queries send, create and cr-create, naive forwarding of queries and replies in the simulation will fail, since the list of potential IS values is used in $\pi^\dagger$ to compute the MAC values.

For example, consider the case where $\hat{A}$ and $\hat{B}$ engage in two pairs of sessions, where in both $\hat{A}$ is the initiator and $\hat{B}$ the responder. If $\mathcal{A}_{\pi^\dagger}$ does not send the final message of the first session $\hat{A} \to \hat{B}$, then the messages in the second session will differ. Indeed, $\pi^\dagger$ is precisely designed to detect this message loss and update anyway, so it will send a MAC with the new state; while $\pi^\star_{\mathsf{MAC}}$ will not and therefore will use the old state.

But this is not a problem, because while simulating $\mathcal{A}_{\pi^\dagger}$, $\mathcal{A}_{\pi^\star_{\mathsf{MAC}}}$ can make an additional $\mathsf{send}(\hat{B},\cdot)$ to $\hat{B}$ to complete the original session. The update procedures in $\pi^\star_{\mathsf{MAC}}$ and $\pi^\dagger$ are identical, so this causes the states to agree before the construction of the final message.

This will perfectly simulate $\hat{B}$'s reply, as if $\hat{B}$'s state updated because of the $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ variable. It is not difficult to check the other cases, or that the simulation can still work if $\mathcal{A}_{pi^\dagger}$ ever decides to go back and finish a session that $\mathcal{A}_{\pi^\star_{\mathsf{MAC}}}$ automatically finished. The other queries can be simulated in the normal way. This shows that $\pi^\dagger$ is secure in Ω_{AKE}. ■

We now give a symbolic argument for robustness. Let π be a stateless robust protocol, and $\pi^\dagger$ its transform. Let $\#(t)$ denote the number of invocations of the KDF in the construction of a term t.

Lemma 33. *For any network adversary $\mathcal{A}$, if at some time during the execution of $\mathcal{A}$ we have two values $st_1, st_2 \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$, then $\#(st_1) = \#(st_2)$*

Proof: A value st' is added to a non-empty $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ just when a message $g^x, \text{MAC}(st^{\mathcal{R}}_{\hat{B},\hat{A}};\cdot)$ is received; since $st' = KDF(\ldots, st)$ we have that $\#(st') = 1 + \#(st)$. Thus we have the stronger invariant that $\#(st') = 1 + \#(st^{\mathcal{R}}_{\hat{B},\hat{A}})$ at all times, which proves the lemma. ■

Lemma 34. *If $st^{\mathcal{I}}_{\hat{A},\hat{B}} = st_1$ and $st^{\mathcal{R}}_{\hat{B},\hat{A}} = st_2$, then $\#(st_1) \geq \#(st_2)$.*

Proof: Every state update $st \mapsto st'$ increases $\#(st)$ by one; we thus need to show only that $\hat{A}$ performs this update before $\hat{B}$, but this indeed holds by induction on the trace length. If the final event in the trace updates $st^{\mathcal{R}}_{\hat{B},\hat{A}}$ then we have equality:

(i) if it was receipt of the third protocol message then $\hat{B}$ receives a message from $\hat{A}$ with state st and updates its state to st
(ii) if it was receipt of the first protocol message then the same still holds.

Thus for all traces we have the inequality. ■

Corollary 35. $\#(st^{\mathcal{I}}_{\hat{A},\hat{B}}) - \#(st^{\mathcal{R}}_{\hat{B},\hat{A}}) \in \{0,1\}$

Proof: We need only show that $\hat{A}$ cannot update twice without an intermediate update by $\hat{B}$; the result then follows since each responder update restores equality. But this holds easily: $\hat{A}$ updates state only when receiving a message m from $\hat{B}$ with the new state, and thus $\#(st^{\mathcal{I}})$ can be at most one ahead of $\#(st^{\mathcal{R}})$ at the time m was sent, and hence certainly no further ahead. ■

Lemma 36. *If $st^{\mathcal{I}}_{\hat{A},\hat{B}} = IS$ at some time t, then either(i) $st^{\mathcal{R}}_{\hat{B},\hat{A}} = IS$ at time t, or (ii) $IS \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ at time t.*

Proof by symbolic argument: Note that state is linear; that is, that there is only one possible value of $st^{\mathcal{R}}_{\hat{A},\hat{B}}$ at a given time. We first claim that the state never repeats; that is, if $st^{\mathcal{I}}_{\hat{A},\hat{B}} = IS$ at times $t_1 < t_2$ then $st^{\mathcal{I}}_{\hat{A},\hat{B}} = IS$ at all times $t_1 \leq t \leq t_2$. This holds by case analysis on the rules which change the state, all of which make the change $st \mapsto KDF(\ldots, st)$ and therefore the number of function symbols in the state term only increases.

Now, let S_n be the statement that the lemma holds for all traces of length $\leq n$. We prove S_n for all n by induction. The base case is trivial; for the inductive case, we need only consider the rules which may change either the $\mathcal{I}$ or the $\mathcal{R}$ state, since if the final rule does not change either state then the induction hypothesis applies to the truncated trace.

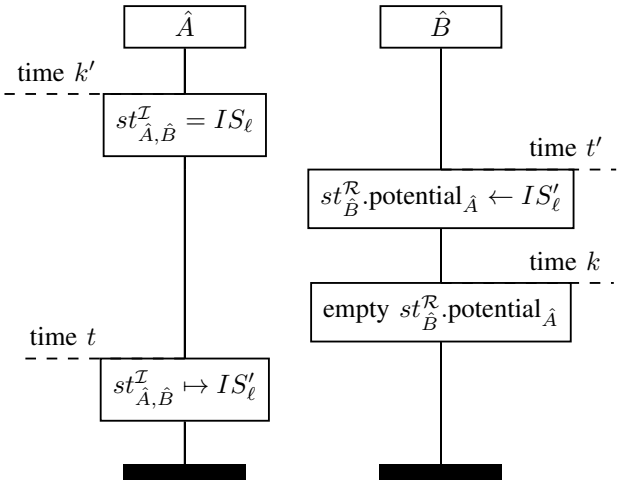


Figure 6: Time ordering of events in the proof of Lemma 36.

Update initiator state. Suppose therefore that we update $st^{\mathcal{I}}_{\hat{A},\hat{B}} : IS_\ell \mapsto IS'_\ell \neq IS_r = st^{\mathcal{R}}_{\hat{B},\hat{A}}$ at time t. By explicit key confirmation we have that $IS'_\ell \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$, added at some time $t' < t$. If it was never removed then we are done, so suppose that it was removed by sending a message m to $\hat{B}$ at some time $t' < k < t$; choose the earliest possible $k > t'$. Finally, since the message causing it to be added must have been sent with state IS_ℓ at some time k', we have $k' < t' < k < t$. The removal can be upon $\hat{B}$'s receipt of either the first or the third protocol message, so there are two cases to consider.

Third. Suppose that m is the third and final protocol message, causing $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$to be emptied as part of the pre-accept computation. We derive a contradiction from Lemma 33: suppose $\#(IS_\ell) = n$, so that $\#(IS'_\ell) = n + 1$. The construction of the third protocol message m occurs only when $\hat{A}$ changes state; by linearity, since $st^{\mathcal{I}}_{\hat{A},\hat{B}} = IS_\ell$ at k' and k it must be so for all times between k' and k. Therefore, m was constructed at $\hat{A}$ before k', say with state st such that $x := \#(st) < n$.

Now, since k was chosen as early as possible, we have that $IS_\ell \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ at time k since it was not removed beforehand. Since m is only accepted if the new state is currently in $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$, we have that $st \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$. By Lemma 33, therefore, $n = \#(IS_\ell) = \#(st) < n$, which is a contradiction.

First. Suppose that m is the first protocol message, causing $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$to be emptied because it is sent with a state $st \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$. Again by Lemma 33, since $IS'_\ell \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ we must have that $\#(st) = n + 1$. This is impossible, since the final event in the trace has $\hat{A}$ update to IS'_ℓ and therefore it sent no messages with such a state.

Update responder state. Suppose that the final rule, invoked at time t, changes the responder state. Let t' be the time of the last change to $st^{\mathcal{I}}_{\hat{A},\hat{B}}$, setting it say to IS_ℓ, so that the induction hypothesis holds at time $t' < t$. Thus we have either $st^{\mathcal{R}}_{\hat{B},\hat{A}} = IS_\ell$ or $IS_\ell \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$. By Corollary 35 we must have that $\#(st^{\mathcal{R}}_{\hat{B},\hat{A}})$ changes from $\#(IS_\ell) - 1$ to $\#(IS_\ell)$, since the corollary holds both before and after the change. The only messages that trigger such a change are sent by $\hat{A}$ with a state st having $\#(st) = \#(IS_\ell)$, and hence since states are linear $st = IS_\ell$. It follows that $st^{\mathcal{R}}_{\hat{B},\hat{A}} = st^{\mathcal{I}}_{\hat{A},\hat{B}}$ after the update, satisfying the conclusion of the lemma. ■

Theorem 37. *If $\pi \in \mathsf{SL}$ is robust then the transformed protocol $\pi^\dagger$ is post-network robust.*

The high-level argument here is as follows. For robustness it suffices to show that if the initiator has some state IS then the corresponding responder either has the same state or derived it in a previous session and did not erase it. Since updates occur upon receipt of a message depending on the previous state, parties can never move more than one update out of sync. We therefore just need to show that required states are not erased by the responder before they are needed. This comes down to a careful case analysis on the triggers for such an erasure.

Proof: If IS(π) is not post-network robust, then by definition this means that there exists some network adversary $\mathcal{A}$ such that, after $\mathcal{A}$ terminates, either(i) the running of $\mathcal{C}$ (the benign adversary that causes a pair of matching sessions between $\hat{A}$ and $\hat{B}$ and then terminates) does not result in $\hat{A}$ and $\hat{B}$ deriving the same session key, or (ii) $\mathcal{C}$ is unable to even create a matching session between $\hat{A}$ and $\hat{B}$. The latter case is ruled out by Corollary 35. The former case is dealt with by Lemma 36: the initial message sent by $\mathcal{C}$ has a state which is already known to $\hat{B}$, and hence by construction the protocol accepts and derives equal session keys. ■