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

***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.

***Index Terms*—Post-Compromise Security, Security Protocols, Key Exchange, Ratcheting, Future Secrecy, Threat Models.**

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.

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

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.*

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.

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 does not learn 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 [26] 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.

Some protocols already maintain "key continuity" in this fashion. Examples include the telephony protocol ZRTP [10] and the instant messaging protocols OTR [6] and its asynchronous successor Axolotl [29]; the latter has been implemented in Signal, WhatsApp[1] and Langley's Pond [28]. 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. We revisit related concepts in Section 2. In Section 3 we discuss the high-level design of security models and how to incorporate PCS. In Section 4 we recall concrete game-based AKE models, and in Section 5 we extend them formally. We discuss related work in Section 6 and conclude in Section 7. We provide proofs for all our formal statements in the full version of this paper [13].

1. Open Whisper Systems originally designed TextSecure, a secure messaging system consisting of a custom key exchange (TripleDH) composed with a ratcheting scheme (Axolotl). The name refers both to the protocol and to the mobile app implementing it. TextSecure-the-app has since been replaced by "Signal", TextSecure-the-protocol renamed to the "Signal Protocol", and Axolotl renamed to the "Double Ratchet". WhatsApp implements the Signal Protocol, and the Signal Protocol uses the Double Ratchet. We discuss this ecosystem further in Section 6.

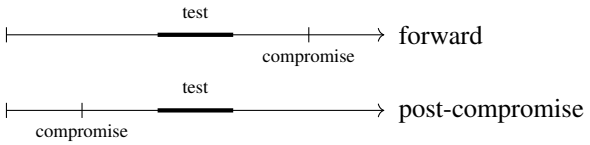


Figure 1: Attack scenarios considered by forward and post-compromise secrecy; "test" refers to the session under attack. Forward secrecy protects sessions against later compromise; Post-compromise secrecy protects sessions against earlier compromise.

2. Related Concepts

There are many techniques for maintaining security in the face of compromise. We take this opportunity to clarify some of the different terminology that has been used.

Ratcheting. The term "ratcheting" has been used previously in the community, but has recently been popularised by Axolotl. It generally describes systems which iteratively apply a one-way function, but within that class often takes different meanings.

The simplest ratcheting design generates a chain $x \mapsto H(x) \mapsto H(H(x)) \mapsto \ldots$, deleting old entries at each stage. This is the approach used by [21] and in the original SCIMP [30]; roughly speaking, it provides forward secrecy in the sense that past values cannot be computed from present ones, as in the first row of Figure 1.

Such designs do not provide security as per the second row of Figure 1, since from x all future values of $H^n(x)$ are easily computable. Thus, Axolotl and the protocols of Section 5.2 additionally mix in in ephemeral values at each stage, so that each update requires knowledge of fresh information. Such systems can also be called "ratchets."

Finally, there are systems which mix in old values but do not ratchet; for example, the construction of [15] to resist compromised RNGs. The guarantees provided by such systems can be incomparable to those above.

Future secrecy. To the best of our knowledge this term is not well-defined; we believe that it is often intuitively used for what we call post-compromise security, but that the details are both crucial and subtle. There is no obviously-canonical way to dualise the definition of forward secrecy at a formal level, though intuitively it means a property as in Figure 1.

Perrin [31] defines "future secrecy" to mean that "a leak of keys to a passive eavesdropper will be healed"—in other words, that an adversary who learns keys but does not interfere with network traffic only learns secret data for a bounded time period. This makes sense intuitively, but a closer look reveals that it is in fact a form of *forward* secrecy. The key insight is that a passive adversary can perform their computations at any point, and so no extra advantage is gained by learning long-term keys before the initial messages. 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.

In more detail, suppose some protocol π has (weak) forward secrecy: no attacker can learn long-term keys and retroactively discover past session keys. Then π already satisfies the above form of future secrecy. Indeed, suppose for contradiction that some passive attacker $\mathcal{A}$ violates the future secrecy of π: it learns the long-term keys, eavesdrops on a particular session s, and subsequently can distinguish the session key of s from random. Since the eavesdropping does not depend on knowing the long-term keys, it is possible to exchange the order of these two operations, defining an attacker $\mathcal{A}'$ which first eavesdrops and then reveals long-term keys. But $\mathcal{A}'$ is then an attacker against forward secrecy, contradicting our assumption.

Healing. ZRTP [10] uses the term "self-healing". Again, we believe that this intuitive notion is challenging to capture formally, and that it is a form of post-compromise security. The difficulty arises when describing what is implied when a protocol is "healed": clearly it is meant that the protocol is secure against certain classes of adversaries, but the precise definition of which attacks fall into the class is not clear.

Bootstrapping. One can view our stronger form of PCS as a form of "bootstrapping", whereby secrecy of one session key K is used to ensure secrecy of its successors. This is a valid intuition; however, it is important to note that it is not sufficient simply to run a secure key exchange protocol which depends on K. The reason is that this protocol must remain secure when composed with the adversary's earlier actions, which does not hold in general. (We show in Section 5.1 that OPTLS is a counterexample.)

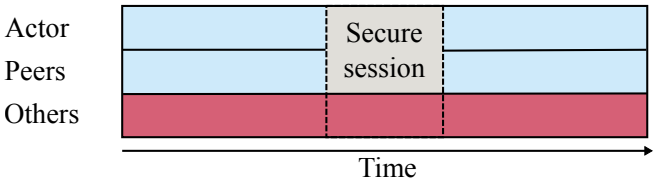
3. Modelling PCS

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.

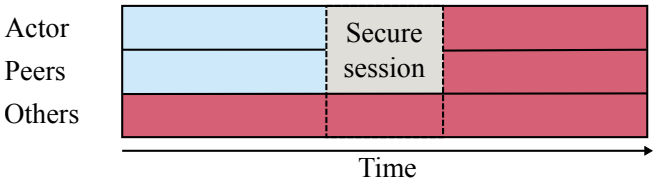
In §4, we instantiate our models concretely in the context of two-party AKE protocols.

The execution of a protocol can be abstractly depicted as in Figure 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.

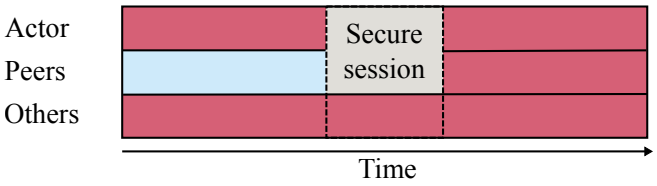
These diagrams are an abstract representation of the model defined formally in Section 5.2, and necessarily do not depict all of its technical details. In particular, we have not shown (i) adversary actions during the secure session, though they are permitted to a limited extent, or (ii) the effects of concurrency, which mean that sessions can cross



(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).

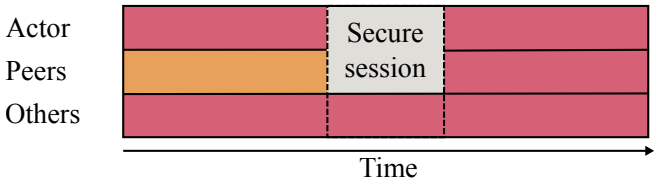
Figure 2: Existing adversary models for security protocols. Figure 2(a) represents a relatively weak model, and Figures 2(b) and 2(c) extend it with PFS and AKC.

the vertical lines.We also do not distinguish in the figures between long-term and ephemeral corruption.

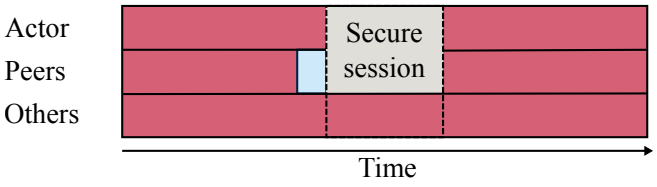
Our high-level goal is to design protocols that achieve security even against a powerful adversary. We always assume that the network is fully untrusted, and that messages may be modified, dropped or inserted in transit. In addition, we can give the adversary access to actors' secret keys or random number 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 2(a), is to grant the adversary access to the long-term keys of any party not involved in the secure session. Security in such a model 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 it by allowing the adversary to compromise all long-term private keys after the end of the attacked



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



(b) **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 3: Our stronger adversary models for security protocols, capturing the notion of post-compromise security.

session [12], as depicted in Figure 2(b). This corresponds to (perfect) forward secrecy, 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 2(c). (Recall that we have fixed "Alice" as the *actor* of the secure session.) This corresponds to the notion of *actor key compromise* (AKC) [2], and was first proposed in the AKE context as *key compromise impersonation* (KCI) [23], in which an adversary uses Alice's private key to impersonate as others to Alice.

Figure 2(c) depicts a modern, strong adversary model. It is clear that there is one more step we could take to further strengthen the model: 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.

We have already seen the following assertion.

Claim (informal)**.** *Post-compromise security is not achievable after* unrestricted *compromise of the intended peer.*

The standard overapproximation is to rule out any compromise at all, as per Figure 2(c). To continue the colour analogy, however, Figures 3(a) and 3(b) 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.

As a concrete running example, suppose Alice and Bob regularly authenticate each other by means of a simple signed DH protocol: Alice generates a random x and sends $\{g^x\}_{sk_A}$; Bob responds symmetrically. Alice's authentication in this protocol relies on sk_A not being compromised: if it is ever leaked, or used to sign a malicious value, then no authentication guarantees can be made. Indeed, the adversary need only obtain a signature on g^z for some z which she knows, and she can then impersonate Alice to Bob. Post-compromise security is a statement about the impossibility of such attacks.

PCS via weak compromise. Figure 3(a) 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.*

One interpretation of this limitation is the assumption of 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.

Another interpretation of the hsm query is as a Bleichenbacher oracle—indeed, Bleichenbacher's original attack precisely allows the attacker to sign (i.e., decrypt) arbitrary data with the server private key, without revealing the key. PCS in this scenario corresponds to preventing precomputation, requiring instead that use of the long-term key depends on data from the secure session. [22] remark that their attacks are more severe when deployed against QUIC precisely because it permits such precomputation.

To continue our signed-DH example, suppose an adversary temporarily gains the ability to generate signatures with sk_A on arbitrary data. She can choose a random number z, compute a signature $\{g^z\}_{sk_A}$, and store it for later use. At any point in the future, she can use that signature to impersonate Alice—even without the ability to generate new signatures. Thus, our simple signed-DH protocol does not admit this form of PCS. We shall see later (Figure 4 and theorem 11) that adding an appropriate nonce challenge-response can prevent this attack.

PCS via state. A more subtle option, as per Figure 3(b), 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 Alice's long-term key. Suppose further that Alice and Bob have already completed a previous session, deriving some secret value v. If Alice's actions in the secure session depend on v—a value from a previous session—then the adversary

cannot necessarily impersonate her. 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.)

To continue our running example, we now have a scenario where the adversary knows sk_A and yet should still be prevented from impersonating Alice to Bob. To do so, we will modify the protocol to use two secrets to derive session keys: not just the long-term key sk_A, but also a secret value derived from the previous sessions. If, after a compromise, Alice and Bob manage to perform a single honest (non-compromised) session, the adversary will not have enough information to compute the subsequent secret values, and therefore cannot compute later session keys. We will cover the details in Section 5.2.

4. Background on AKE Security Models

To concretely instantiate our analysis of PCS in the domain of AKE protocols, we revisit existing AKE analyses. We refer the reader to e.g. [8, 15, 27] for complete definitions.

The majority of security models for authenticated key exchange derive from [4]. Recent notable models are the CK model [12] and the eCK model [27], which introduce ephemeral key reveals under various names and can be extended [16] to cover other properties such as perfect 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. (Further discussion can be found in [1].)

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.

4.1. 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., $\epsilon_{\text{DDH}}(\mathcal{D})$ is negligible in q for any efficient $\mathcal{D}$ where

$$\epsilon_{\text{DDH}}(\mathcal{D}) := \big|\Pr\left[x, y \leftarrow\!\!\$\, \mathbb{Z}_q : \mathcal{D}(g^x, g^y, g^{xy}) = 1\right] - \Pr\left[x, y, z \leftarrow\!\!\$\, \mathbb{Z}_q : \mathcal{D}(g^x, g^y, g^z) = 1\right]\big|.$$

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.

The signature scheme is defined to be EUF-CMA if the probability of the adversary winning the above game is negligible in the security parameter.

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 [11], though none are yet practical.

4.2. Oracles and Queries

We work in the tradition of Bellare and Rogaway [4], 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

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 §5.1
$\mathsf{session\text{-}key}(\hat{A}, i)$	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 $\mathcal{A}$.

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 "queries". Many different queries have been studied [5, 12, 27] 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 [15]. 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, and take on the role of either $\mathcal{I}$nitiator or $\mathcal{R}$esponder. 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 *[16] 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 [8, 9], 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*

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

of $\mathcal{A}$ is negligible in the security parameter.

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 4 is secure in the model X_{basic} under the DDH assumption.*

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

5. PCS in AKE

This section will instantiate the informal definitions and claims from Section 3 using the game-based security models described in Section 4. We first consider the case of weak compromise before returning to the full access case.

5.1. Weak Compromise

In order to model limited access to peers' identity keys before the test session, we add a new query $\mathsf{hsm}(\dots)$ to the model. This query will effectively act as a limited version of $\mathsf{corrupt}$ 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 $\mathsf{hsm}(A; \dots)$ to return $\mathcal{I}(A, \mathsf{sk}_A, \dots)$, 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 $\mathsf{corrupt}$.

Definition 10. *Let* $\mathsf{pcs\text{-}hsm}$ *denote the trace predicate that*

(i) for all queries $\mathsf{hsm}(x, \dots)$, $x = s_{peer}$; and

(ii) all queries $\mathsf{hsm}(x, \dots)$ 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 in this model 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 $\mathsf{corrupt}$ 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.

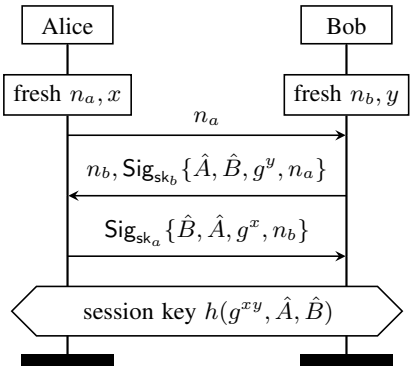


Figure 4: A simple signed challenge-response DH protocol.

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 4, 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 4, 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 specification of TLS 1.3, the upcoming standard for secure communication over the Internet). One proposal, OPTLS [26], has servers generate and sign temporary public keys that are then used for a signature-free handshake. (Google's QUIC also has this design.) After the client sends the initial hello message and DH share, the server responds with its hello, a server DH share g^s, a signature on g^s, and a MAC computed over some messages. Because servers typically possess certified RSA signature keys, OPTLS allows for the signing of g^s with the server's signature key. Thus, the server does not need to use a DH certificate. In one mode, the signature on g^s covers both g^s and a validity period through which a client can accept server authentication based on g^s. This mode saves the cost of a per-session signature, but endows the server with the power of certification abilities. This means that a signed g^s is all that is required to impersonate the server. In the extreme case, an attacker with temporary HSM access could impersonate as a server supporting TLS 1.3, even when the real server does not even support it.

The intuitive property of the protocol in Figure 4 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 [17, 25], 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, even though it provides forward secrecy with an ephemeral DH exchange, and even if used with a TPM.

5.2. Full Compromise

Theorem 11 shows that a form of PCS can be achieved 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, an 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.

As discussed in Section 3, it is not hard to show that stateless protocols cannot meet this notion of security: an adversary can steal Alice's keys and then perform the same computations as Alice. We show now that stateful protocols can resist such attacks.

The intuition is as follows. Every time Alice and Bob complete a session, they derive two secret, independent values from the randomness, protocol state and messages: the session key and a token. The session key is the output of the protocol, as usual. The token is used in the key derivation for the *next* session completed with the same peer; thus, an adversary who wishes to simulate Alice's computation must be able to derive the token in addition to all the other secret inputs. We call such a design a *key refreshing protocol*.

Since a new token is generated every session, even a single uncompromised session implies secrecy of future keys; moreover, if the token is derived in the same manner as the session key, then it is afforded the same protection and can only be learnt by revealing an agent's state. This is the core of our protocol transformation, though there are various subtleties that we discuss later.

Modelling. Of course, this argument is informal. To make it precise we must define a formal security model, which specifies exactly which actions the adversary is allowed to take. We proceed as follows. First, we define a minimal model (KE-PCS) which *only* allows the attacker to make PCS attacks, and we formally prove 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$^{\text{w}}$, eCK$^{\text{w}}$-PCS), and second to a strong model (Ω_{AKE}) for the class of all AKE protocols. Finally, we show that our models are achievable by specifying protocols that meet them.

Recall that cr-create represents creation of a session with chosen randomness, so is a more powerful version of the randomness query, and that we model full state compromise through randomness queries against all past sessions.

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

(i) $s_{actor} = s'_{actor}$ *and* $s_{peer} = s'_{peer}$,
(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* $F = P_1 \wedge P_2$ *as follows.*

(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 refreshed*

The model KE-PCS captures precisely (and only) the concept of full PCS, the blue area in Figure 3(b): if the adversary compromises the long term keys of the peer before the test session starts, and the test session is not refreshed, then it can impersonate the peer. (In this case there is no origin-session for the messages received by the test session.)

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 [16]. 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 [27] 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 [15] 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 $\Omega_{AKE\cap ISM}$ [15] and Ω_{AKE} are defined by their freshness predicates as follows.*

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

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 refreshed.

TABLE 3: Clauses of the freshness predicates for our AKE models. The predicates with superscript eCK$^{\mathrm{w}}$ and ISM come from [16] and [15] 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 $F_3^{\mathsf{eCK^w}}$ 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.

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 peer), and the key of s_{peer} has been revealed, then there must have been a "blue area" or key refreshing session allowing security to be re-established.

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.

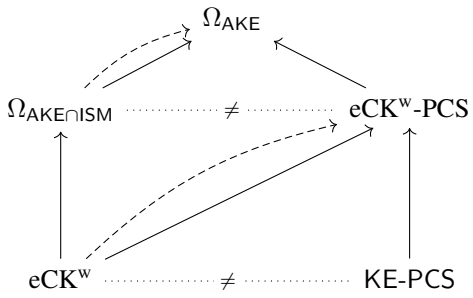


Figure 5: Relationship between the various models we define in this section. $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ and eCK$^{\omega}$ are from [15] and [16] respectively. We propose the other three modules in this work. Solid arrows denote inequality of security models ($\leq_{sec}$, Appendix B), dotted 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.

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 5.1 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 key refreshing 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.*

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 6(a). In Figure 6(b) we show an example execution that illustrates the robustness mechanism. We refer the reader to the full paper [13] 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 security of the modified protocol $\pi^{\dagger}$ follows from adding an intermediate secret in the KDF, and continuously updating it as sessions take place[2]. Our construction additionally depends on the (weak) perfect forward secrecy of π. Because $\pi^{\dagger}$ also has this property, the attacker cannot compute the secrets added in the refreshed session, even if it has one of the long-term keys.

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 full paper [13]. 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 refreshed 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 a direct reduction.

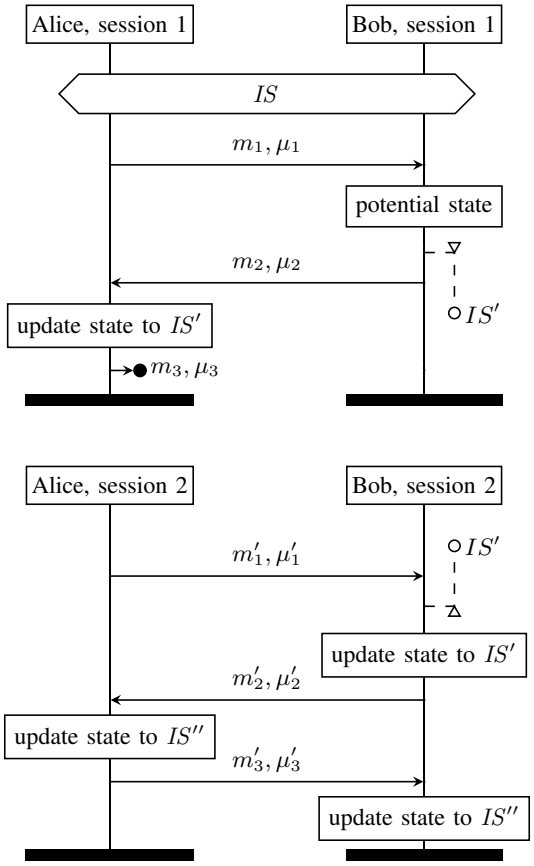
Results in the computational setting are generally negative, in the sense that they prove that the adversary cannot distinguish between certain values. Robustness is not a property of this form, and therefore the standard game-hopping techniques are not immediately applicable. We

2. In fact, protocols transformed from eCK$^{\mathrm{w}}$ are secure in a stronger model than eCK$^{\mathrm{w}}$-PCS: the 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.

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. We define $\pi^\dagger$ as a modification of π as follows.

- For each agent $\hat{B}$ we define new global state variables $IS^{\hat{B},\hat{A}}$ and $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ (for each communication partner $\hat{A}$). The first stores a single value and the second a possibly empty list of values. Initially, $IS^{\hat{A},\hat{B}}$ is set to g^{ab} for all $\hat{A}, \hat{B}$.
- 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 $is \in st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$ 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 $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}} \leftarrow \emptyset$, and continues to the next step.
  - Otherwise, $\hat{B}$ `rejects`.
- The new value $IS^{\hat{B},\hat{A}}_{\text{new}} \leftarrow \text{KDF}(\sigma, IS^{\hat{B},\hat{A}}, 1)$ is appended to $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}}$.
- 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 $IS^{\hat{A},\hat{B}} \leftarrow IS^{\hat{A},\hat{B}}_{\text{new}} = \text{KDF}(\sigma, IS^{\hat{A},\hat{B}}, 1)$, 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 $IS^{\hat{B},\hat{A}} \leftarrow IS^{\hat{B},\hat{A}}_{\text{new}}$ and $st^{\mathcal{R}}_{\hat{B}}.\text{potential}_{\hat{A}} \leftarrow \emptyset$.

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



(b) Example trace of a transformed protocol. 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 6: Two-round, robust PCS transform.

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.

6. Related Work

There is relatively little work on related phenomena. The concept of "key continuity" appears in various Internet Drafts [10, 20], 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 [18], ZRTP [10], and even many SSH [32] implementations via the known-hosts feature. However, existing analyses of these protocols do not cover this aspect.

Bowers et al. [7] 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 [21] 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 [15] 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 [3]. State is of course used *within* sessions—for example, ACCE [24] provides security to the message channel in the sense of stateful length-hiding authenticated encryption. It is also used heavily at the application layer, particularly in the web context.

TextSecure/Axolotl. The Axolotl (now Double) ratchet works roughly as follows. As Alice and Bob communicate, they share a dynamically changing "root key" rk, similar to our intermediate secrets. Each sent message is accompanied by a DH share, and the root key is updated as $\mathsf{rk} \leftarrow \mathrm{KDF}(H(g^{x_i y_j}, \mathsf{rk}))$, where g^{x_i} and g^{y_j} are the most recently sent DH shares. Messages are themselves encrypted with keys derived from the most recent root key, which acts as synchronised state; as such, Axolotl's design includes a PCS-like mechanism.

The (recently discontinued) TextSecure messaging app used Axolotl, and one might therefore expect it to achieve PCS. It did not, however, because of its implementation of multi-device messaging: TextSecure shared long-term keys across all devices but maintained a separate Axolotl chain for each, allowing an adversary to impersonate Bob by claiming to be a previously-unseen device after a compromise.

In more detail, the design of the protocol was that each device on a given user's account stored a copy of the user's long-term key, but otherwise did not share state. For Alice to send a message to Bob she would send it separately to each of Bob's devices. Thus, an attacker who learned Bob's long-term key could register a new device on Bob's account, and Alice would accept the new device as valid, since the correct identity key was used to set it up. Hence, if Bob's key is compromised then he could be impersonated to anybody. In particular, relating to the definition of PCS, the scenario would be the following: First, the attacker learns Bob's (long-term) identity key. Next, Alice and Bob have a key refreshing (honest) session. If PCS were to be achieved, the attacker would now be locked out. However, the attacker can now register a new device for Bob using only the identity key, and this enables the attacker to impersonate Bob again even though the session under attack has been refreshed. Thus, PCS is not achieved. As a consequence, it is not 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.

A previous analysis of TextSecure [19] reported an unknown key share attack. Because the analysis in [19] does not consider any security after the peer's compromise, it does not cover PCS.

7. 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 module to store keys can allow the effects of a 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 key refreshing 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.

In addition to the long-term possibilities of extending our work to other domains, there are other direct avenues of future work:

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 [31] 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] David Basin and Cas Cremers. "Know Your Enemy: Compromising Adversaries in Protocol Analysis". In: *ACM Trans. Inf. Syst. Secur.* 17.2 (Nov. 2014), 7:1–7:31. ISSN: 1094-9224. DOI: 10.1145/2658996. (Visited on 04/11/2016).

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

[3] 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. DOI: 10.1145/1180405.1180452. (Visited on 11/10/2015).

[4] Mihir Bellare and Phillip Rogaway. "Entity Authentication and Key Distribution". In: *Advances in Cryptology – CRYPTO '93*. LNCS 773. Springer Berlin Heidelberg, Jan. 1, 1994, pp. 232–249. DOI: 10.1007/3-540-48329-2_21. (Visited on 05/21/2014).

[5] 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: Springer-Verlag, 1997, pp. 30–45.

[6] 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. DOI: 10.1145/1029179.1029200. (Visited on 06/22/2015).

[7] 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.

[8] Christina Brzuska et al. "Composability of Bellare-rogaway Key Exchange Protocols". In: CCS. ACM, 2011, pp. 51–62. DOI: 10.1145/2046707.2046716. (Visited on 07/27/2015).

[9] 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).

[10] 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).

[11] 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).

[12] Ran Canetti and Hugo Krawczyk. "Analysis of key-exchange protocols and their use for building secure channels". In: *EUROCRYPT*. Vol. 2045. LNCS. Springer-Verlag, 2001, pp. 453–474.

[13] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. *On Post-Compromise Security*. Cryptology ePrint Archive, Report 2016/221, 2016. URL: http://eprint.iacr.org/2016/221.

[14] Cas 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] Cas Cremers and Michele Feltz. *On the Limits of Authenticated Key Exchange Security with an Application to Bad Randomness*. Cryptology ePrint Archive, Report 2014/369, 2014. URL: https://eprint.iacr.org/2014/369.

[16] 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. DOI: 10.1007/s10623-013-9852-1.

[17] 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).

[18] 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. DOI: 10.1145/1102199.1102216. (Visited on 01/30/2014).

[19] Tilman Frosch et al. *How Secure is TextSecure?* 904. 2014. URL: http://eprint.iacr.org/2014/904 (visited on 11/07/2014).

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

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

[22] Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. "On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 V1.5 Encryption". In: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. ACM, 2015, pp. 1185–1196. DOI: 10.1145/2810103.2813657. (Visited on 10/19/2015).

[23] 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. Springer-Verlag, 1996, pp. 36–49. URL: http://dl.acm.org/citation.cfm?id=647093.716554 (visited on 02/09/2016).

[24] 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).

[25] 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).

[26] 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).

[27] B.A. LaMacchia, K. Lauter, and A. Mityagin. "Stronger Security of Authenticated Key Exchange". In: *ProvSec*. Vol. 4784. LNCS. Springer, 2007, pp. 1–16. DOI: 10.1007/978-3-540-75670-5_1.

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

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

[30] Vinnie Moscaritolo, Gary Belvin, and Phil Zimmermann. *Silent Circle Instant Messaging Protocol Specification*. Dec. 2012. URL: https://silentcircle.com/scimp-protocol (visited on 07/15/2015).

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

[32] 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 4, 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 4.

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 [15], 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.

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

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

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

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

A similar impossibility result, analogous to [15, 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 refreshed. 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 refreshed.*

(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 refreshed.*

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$^{\text{w}}$-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.
Full Paper: Robustness and PCS Transform

In the full version of our paper [13], we prove the following impossibility result: no stateless protocol can be secure in KE-PCS (Theorem 14). This motivates our generic transformations by showing that it is necessary to leave the class of stateless protocols if one wants to achieve our strong notion of PCS. This leads to our generic transformation to turn a protocol π in the class ISM into a protocol $\pi^\star$ not in ISM.

First, for simplicity and intuition, we provide a generic construction and proof of security in the case of π being a one-round protocol. We show that $\pi^\star$ achieves security in strong PCS models, but is not robust in the sense that parties can fall permanently out of sync with a unreliable network. Indeed, we prove that this is always the case with a one-round protocol and therefore a trade-off between security and efficiently must always occur.

However, the proof technique for our one-round protocol transform allows us to then provide a security proof for a two-round protocol. Our two-round protocol transform is more complicated in that it involves MACs and potential vs actual intermediate secrets in memory, but the basic intuition of sharing and updating an intermediate secret is the same. We prove that this protocol can be robust and achieve PCS.