How fast do you heal?
A taxonomy for post-compromise security in secure-channel establishment

Olivier Blazy
LIX, CNRS, Inria, École Polytechnique, Institut Polytechnique de Paris, France

Ioana Boureanu
University of Surrey, Surrey Centre for Cyber Security, UK

Pascal Lafourcade
LIMOS, University of Clermont Auvergne, France

Cristina Onete
XLIM, University of Limoges, France

Léo Robert
LIMOS, University of Clermont Auvergne, France

Abstract

Post-Compromise Security (PCS) is a property of secure-channel establishment schemes, which limits the security breach of an adversary that has compromised one of the endpoint to a certain number of messages, after which the channel *heals*. An attractive property, especially in view of Snowden's revelation of mass-surveillance, PCS was pioneered by the Signal messaging protocol, and is present in OTR. In this paper, we introduce a framework for *quantifying and comparing* PCS security, with respect to a broad taxonomy of adversaries. The generality and flexibility of our approach allows us to model the healing speed of a broad class of protocols, including Signal, but also an identity-based messaging protocol named SAID, and even a composition of 5G handover protocols.

1 Introduction

Secure-channel establishment is a cornerstone of modern communication. It allows users to exchange messages whose confidentiality and authenticity are guaranteed, even with respect to potential Person-in-the-Middle attackers. Authenticated Key-Exchange (AKE) protocols have, for decades now, enabled such channels to be established: for Internet browsing (TLS protocol), for mobile networks (AKA protocol), for secure remote access to another machine (SSH), and so on.

The revelations of Edward Snowden, who exposed the reality of mass surveillance by security agencies such as the NSA or GCHQ, were a boost to the widespread deployment of secure channels. It is now confirmed that powerful adversaries can, and do, fully corrupt the private information stored on a targeted device, thus learning most (if not all) of its secrets. Even then, secure channels open *prior* to the adversary's intrusion can still preserve confidentiality and authenticity if Perfect Forward Secrecy (PFS) is guaranteed. Unfortunately, however, all sessions *following* the party's compromise will no longer guarantee either confidentiality or authenticity.

The lack of future security is particularly problematic for secure channels that are meant to last for a long time, such as those generated by asynchronous messaging applications. Say that a civilian, Alice, has a journalist friend, Bob, with whom she communicates via a secure messaging application. While abroad, Alice receives sensitive documents from a whistleblower, whose request is that she sends them to Bob. She messages Bob about them. But as she travels back home, Alice's phone might be compromised at border control. At this point, she would like to have three guarantees: her past communication with Bob is secure; no one can impersonate her to bait Bob; and in a little while, she will be able to resume talking to Bob without (for instance) destroying her phone.

The PFS of the channel guarantees the first of these requirements. For the second and third, Alice requires a property pioneered by Marlinspike and Perrin in the context of the Signal protocol [21], called *Post-Compromise Security* (PCS) by Cohn-Gordon *et al.* [15]. This attractive feature implies that the secure channel established in Signal by Alice and Bob can repair (or "*heal*") its security, even after a full compromise.

But of course, Alice wants to know: how fast will the channel's security return, and under what conditions? Cohn-Gordon *et al.* [13] showed that in the original Signal protocol, Alice can recover security after she and Bob have switched speakers (*i.e.*, exchanged sender/receiver roles) twice in the conversation. So, if Bob was the current sender when Alice's phone was compromised, then Alice must first send (at least) a message, and wait for Bob's reply before they are safe again. All messages sent between the moment of corruption and Bob's second reply are compromised by the attacker.

Even more problematic is the case in which the attacker uses the data recovered from Alice to insert itself into the communication, choosing whether it wants to just impersonate Alice to Bob, impersonate Bob to Alice, or both, and set itself up as a permanent Person-in-the-Middle between them. For active attackers, Alice's conversation with Bob *never* heals.

Hailed as a revolutionary design in secure-channel establishment, Signal was used as a basis for group messaging schemes, such as ART [14] and MLS [5]; the protocol can also be used directly if group messaging is implemented as a composition of pairwise secure channel between all the participants. Other messaging protocols, such as OTR [9], Matrix [1], Wire [20], also guarantee some measures of PCS.

To improve the healing speed of Signal, Blazy *et al.* [7] used identity-based cryptography and introduced SAID, a protocol which provides much better security against active attacks. As SAID relies on a different paradigm than Signal, it is hard to compare their respective security levels. Yet, at a high level both protocols guarantee post-compromise security – so are they perhaps equivalent?

In our example, the healing speed makes a huge difference to Alice. She would moreover ideally like to know that healing depends entirely on her (rather than, say, on Bob returning online). Finally, from a designer's point of view it is crucial to understand how different protocols handle different adversaries: would the attacker only require short-term (potentially more vulnerable) values, or does it need long-term secrets?

Our contributions. We propose a *metric* allowing to assess and compare the post-compromise security of apparently-incomparable protocols, such as Signal and SAID. Our aim is to quantify the healing speed and healing conditions of a scheme with respect to various *classes* of attackers.

More precisely, our contribution is threefold: we formally define a broad category of two-party protocols that enable key-evolution (which we call SCEKE, for Secure-Channel Establishment schemes with Key-Evolution); we then define a framework, consisting of a *post-compromise security (PCS) metric* and several classes of adversaries, varying in strength and abilities; and we provide a comparison of three SCEKE protocols, thus exemplifying our framework.

For this, we introduce a taxonomy of adversaries, in terms of 3 characteristics: their *access* (is it a trusted party or not), their *power* (active or passive), and their *reach* (which values does it compromise?). A weak adversary may only be able to compromise stage-specific values (which are always in memory). A very strong adversary might be a trusted party (like Signal's credential server), able to actively hijack sessions and fully compromise all the data belonging to a party.

Our PCS adversaries attack SCEKE protocols, in which channel keys are used for a short time (*i.e.*, during a *stage*). We index stages by pairs of positive integers (x,y), and consider an evolution of stages that is either *horizontal* (from stage (x,y) to $(x+1,y)$), or *vertical* (from stage (x,y) to $(1,y+1)$). Stages with same y are said to be on the same chain.

PCS adversaries will target a specific channel (also called message) key, having compromised an endpoint to that channel. Our metric measures the *number of messages* required, per message-chain, for the security of the channel to heal after this corruption. Thus, a protocol is $(\infty,1)$-PCS resistant with respect to some class of adversaries if the honest parties lose channel security for all the stages obtained through horizontal evolution, and at most 1 stage obtained through vertical evolution, starting from the last stage (x^*,y^*) at which the adversary compromised either endpoint. Optimal healing corresponds to $(1,0)$-PCS security, while the worst healing is (∞,∞)-PCS security, *i.e.*, the protocol's security never heals.

To showcase the broad reach of our metric, we use it to compare 3 schemes that would otherwise be hard to compare: the PKI-based Signal asynchronous messaging protocol (analyzed by Cohn-Gordon *et al.* [13]), the identity-based SAID asynchronous-messaging protocol [7], and the 5G handover protocols in mobile networks [2,3]. For the latter protocol, we are the first (to our knowledge) to model and analyze the post-compromise security afforded by sequential compositions of handovers. We also show how to easily tweak 5G handovers to obtain much faster healing. Our results are summarized in Fig. 4. Our framework can also be used to analyse the security of many other PCS protocols.

Related work. Provable-security analyses of known protocols, such as those of Signal [13] and SAID [7] are cornerstones to our work, and enable us to show how our framework compares with existing results. This is one of the reasons those two protocols were chosen. However, in our work we go beyond current results, both in terms of scope and of provable results.

Although comparatively infrequent, taxonomies and metrics exist and are very useful in cryptography. An eminent example is the taxonomy of privacy notions by Pfitzmann and Hansen [22], which ranks and classifies subtly-different terms referring to user privacy (*e.g.*, anonymity, privacy, unlinkability, undetectability, etc.). We choose to focus on the property of channel security in the context of a particular type of scheme; moreover, while we classify attacks by three *types* of parameters, our taxonomy focuses on a precise *quantification* of healing speed, which is out of scope for [22].

Our methodology better resembles taxonomy efforts such as [17,19], whose purpose is to categorize security definitions in information-flow, or electronic voting, respectively. Our work, however, focuses on a very different type of protocol than in these two fields; moreover, we use a provable-security, rather than a formal-methods approach.

The mechanics of our model resemble those of Fischlin and Günther [18], which extend prior work [6,12] to complex, multi-stage key-agreement. Our main security notion (Post-Compromise Security), however, is atypical for [18]; moreover, we are the first to define the generic notion of SCEKE protocols. A parallel line of work recently introduced by Brzuska *et al.* [10] analyzes, in a compositional framework (using state-separating proofs [11]), the security of the multi-party asynchronous messaging protocol MLS. Unlike that technique, ours is not composable[1], but unlike [10] we do take into account authentication, which is crucial in the case of active adversaries.

Finally, we note that our main contribution in this work is the taxonomy of attackers and quantification of PCS-security.

[1]This is, technically speaking, because we quantify PCS-security within the *winning conditions*.

This is why we carefully chose only three protocols (with apparently incomparable degrees of PCS) to analyze: Signal, SAID, and the 5G Handover protocols. We discuss how our framework applies to other protocols below.

Other protocols. Although we choose to showcase our metric by means of the three protocols cited above, our framework can be applied to other PCS-providing protocols, such as OTR, Matrix' Olm protocol for 2-party rooms, and Wire. Signal ratchets are actually a combination of OTR and SCIMP ratchets[2]. Notably, the former provides PCS security. However, note that OTR's focus is privacy, not necessarily (PCS-)security, and thus limits and encrypts any explicit long-term-authentication steps. This gives it a relatively weak security in our framework when we consider active adversaries, but interestingly provides less advantages for insider attackers. An interesting future research is how to optimally balance the kind of privacy desired by OTR and its healing speed.

The Olm protocol used by Matrix resembles Signal (with differences regarding the type of keys used, signed or unsigned) and would provide similar metric results in our framework – which is why we do not treat it. On the other hand, Wire is more complicated to analyse. Although the core protocol relies on an independent implementation of Signal, its use of cookies and access tokens for authentication and synchronization complicates matters, particularly with respect to powerful adversaries such as insiders. Moreover, the ability to have multiple synchronized devices raises the questions of modelling individual-device compromise and device revocation, for which we would need an extended framework, akin to what is needed to capture MLS security (see below).

A limitation of our approach is that we only consider two-party protocols: as such, even if our *taxonomy* of adversaries is easily extendable to multi-party schemes, such as ART and MLS [5, 14], our *metric* is not. A particular difficulty with extending our framework to multiple parties is the dynamic addition and removal of participants. In two-party schemes, we have two types of evolution, which correspond –roughly– to one, or the other participant's messages. In that case, our metric quantifies the response to the question: if Alice is compromised, after how many of her, and Bob's messages will the channel security heal? However, when we have a dynamically-adaptable set of parties, we would need to account – not only how many turn-switches there are between Alice and non-Alice participants, but also over added and removed users. It is not immediately apparent how best to achieve this, which is why we leave the extension of our metric to multi-party protocols as future work.

2 Our PCS metric for SCEKE protocols

Our framework applies to a generalization of two-party secure-channel establishment, which features key-evolution. We call such protocols Secure-Channel Establishment schemes with Key-Evolution (SCEKE), and emphasize that, while post-compromise security (PCS) is relevant to long-lived secure channels, it can also be an attractive property for short-lived channels whose keys evolve from (or depend on) each other.

2.1 Definition of SCEKE Protocols

SCEKE schemes allow two parties, Alice and Bob, to initially establish a secure channel (by agreeing on some initial key-material), and then preserve the security of that channel over a long period of time by sequential evolutions of the key material, meant to ensure two properties:

PFS: If a user or an instance is fully corrupted at a given moment, all keys established prior to that corruption remain secure;

PCS: Even if a user or instance is fully corrupted at some moment, thus breaking channel security, that security will return after a given, finite interval.

Our metric measures the interval required for security to return, with respect to several classes of adversaries (Sec. 2.3). We begin by formalizing the syntax for SCEKE protocols.

Participants. We consider schemes featuring participants of two types: parties P making up a set $\mathcal{P}$, and a super-user $\hat{S}$ playing a special part in the protocol (like the registration server used in Signal or the key-derivation center present in identity-based infrastructures). Channel-establishment takes place between two parties (rather than a party and $\hat{S}$), which compute keys and have them evolve.

A SCEKE scheme is initiated by means of a setup algorithm, run by one or more parties, which yield a number of private and public parameters: the super-user $\hat{S}$ is associated with the tuple $(\hat{S}.\mathsf{sk}, \hat{S}.\mathsf{pk})$, while each party P retains identity-bound credentials $(\mathsf{ik}_P, \mathsf{ipk}_P)$. Each of these keys could be a concatenation of credentials, or – if absent – could be void (denoted by $\perp$).

Setup precedes user registration, during which parties P register with the super-user, and $\hat{S}$ builds a database with entries indexed by *unique* party identifiers P. The contents associated to each party ID are protocol-specific and is used during secure-channel establishment; exploiting registration leads to a particularly strong type of attack, performed by an *insider* adversary.

Sessions, instances, and stages. After registration, parties can run protocol *sessions* with each other. Following [6, 12] formalizations, a protocol *session* occurs between two party-*instances*. The i-th instance of P is denoted π_P^i.

Each instance must run three types of steps during a protocol session: an initial, one-time initialisation step; a recurring message-sending step, occurring every time the instance *sends* an (encrypted) message to its partner; and a recurring receiving step, corresponding to an instance *receiving* (and retrieving) a message from its partner. The sending and receiving

[2]See https://signal.org/blog/advanced-ratcheting/.

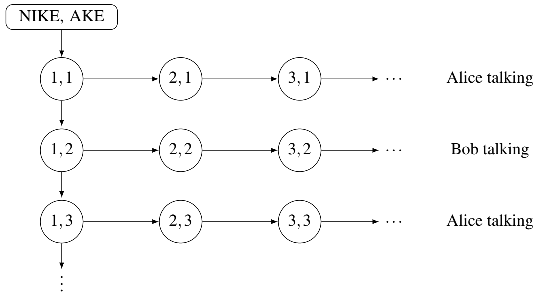

Figure 1: Stage evolution in asynchronous messaging. The protocol initiator begins at stage $(1,1)$ and will continue to send messages along the first horizontal chain. Bob's first reply comes at stage $(1,2)$.

steps depend on a crucial notion, that of *stage*.

In SCEKE protocols, messages are encrypted and authenticated before sending, using one or more *message keys*. A stage corresponds to a (protocol-specific) number of messages associated with the same message key. When that key evolves, we have moved on to the next stage. The first stage is indexed $(1,1)$ and corresponds to message key $\mathsf{mk}^{1,1}$; generically, stages are indexed (x,y) with $x,y \geq 1$, with keys evolving through either *horizontal* or *vertical* evolution, as follows.

Horizontal evolution: Stage (x,y) turns into $(x+1,y)$. This evolution provides weaker security.

Vertical evolution: Stage (x,y) turns into $(1,y+1)$ (we "reset" the x value). This evolution provides stronger security.

As we capture generic key-evolution, depicted in Fig. 1, our definition of stages is intentionally vague. We do, however, require that honest parties always evolve "forwards", that is, always increasing either the x or the y value.

Formalization. More formally, instance π_P^i of parties $P \neq \hat{S}$ keeps track of the following attributes:

pid**:** partner identifier for the session, denoted π_P^i.pid.

sid**:** session identifier π_P^i.sid: an evolving set of instance-specific values.

stages**:** a list of tuples (s,v), of stages $s=(x,y)$, with values $v \in \{0,1\}$ indicating whether a message was received $(v=1)$ or not $(v=0)$. By abuse of notation we write $s \in \pi_P^i$ if, and only if, $(s,v) \in \pi_P^i$.stages.

*Tr***:** transcript $\pi_P^i.T$, indexed by stage s describing all data sent or received for this stage. We denote $\pi_P^i.T[s]$.

rec**:** a list of subsets π_P^i.rec, indexed by stage s and indicating messages and metadata received, in order. A special symbol $\perp$ is used for sending stages.

var**:** a set π_P^i.var of ephemeral values used to compute stage keys, indexed by stage. If a value is used for more than one stage, it will appear under every single stage that it is required for.

Definition 1 (SCEKE Protocol) *A Secure-Channel Establishment protocol with Key-Evolution (SCEKE) is a tuple of five algorithms and two interactive protocols:* $\mathtt{SCEKE} = (\mathtt{aSetup}, \mathtt{aKeyGen}, \Pi_{\mathtt{UReg}}, \Pi_{\mathtt{Start}}, \mathtt{aSend}, \mathtt{aReceive}, \mathtt{aRGen})$*:*

$\underline{\mathtt{aSetup}(1^\lambda) \to (\hat{S}.\mathsf{sk}, \hat{S}.\mathsf{pk}, \mathsf{pparam})}$ *: outputs the public/private long-term keys of super-user $\hat{S}$ and the public system parameters* pparam *implicitly taken in input by all other algorithms.*

$\underline{\mathtt{aKeyGen}(1^\lambda) \to (\mathsf{ik}, \mathsf{ipk})}$ *: run by a party P to output public/private long-term credentials* $(\mathsf{ik}, \mathsf{ipk})$*, used at registration (and perhaps further). Either key could be set to a special symbol $\perp$.*

$\underline{\Pi_{\mathtt{UReg}}(P, \hat{S}) \to (\{\mathsf{sk}, \mathsf{pk}\}, b)}$ *: an interactive protocol run by party P and super- user $\hat{S}$. The latter outputs a bit b (set to 1 for a successful registration), while the former outputs public/private credentials* $(\mathsf{sk}, \mathsf{pk})$ *for $\hat{S}$. The super-user keeps track of a registration database.*

$\underline{\Pi_{\mathtt{Start}}(P, \mathsf{role}, \mathsf{pid}, \hat{S}) \to (\pi_P^i, b)}$ *: run interactively between P and super-user $\hat{S}$, so as to create an instance of P meant to be talking to an instance of* pid*, such that P has either the role of initiator or responder. If successful, $\hat{S}$ outputs b, while P outputs a handle π_P^i on its i-*th *instance. Some initial key material might be computed during this phase (like a master secret).*

$\underline{\mathtt{aSend}(\pi_P^i, s, M, AD) \to (\pi_P^i, C, AD^*) \cup \perp}$ *: on input a state instance π_P^i, a stage s, a message M, and associated data AD, the algorithm outputs an updated state of the instance π_P^i, a ciphertext C, associated data AD^* or a symbol $\perp$.*

$\underline{\mathtt{aReceive}(\pi_P^i, s, C, AD^*) \to (\pi_P^i, M, AD) \cup \perp}$ *: on input an instance π_P^i, a stage s, a ciphertext C, and associated data AD^*, it outputs an updated state of the same instance π_P^i, a message M and some (possibly transformed) associated data AD, or symbol $\perp$.*

$\underline{\mathtt{aRGen}(1^\lambda) \to (\mathsf{rchk}, \mathsf{Rchpk})}$ *: outputs a public/private key-pair used to refresh keys. We call these* ratchet keys*, though they are more generic than the original asymmetric-messaging concept. Either key could be a special symbol $\perp$.*

Protocol correctness relies on matching conversation (intuitively, the partnering of instances running a session).

Definition 2 (Matching conversation) *Let $\mathtt{SCEKE}$ be a SCEKE protocol, and A, B two parties with instances π_A^i and π_B^j respectively. π_A^i and π_B^j have* matching conversation *if and only if $\pi_A^i.\mathsf{sid} = \pi_B^j.\mathsf{sid}$ and $\pi_A^i.\mathsf{pid} = B$ and $\pi_B^j.\mathsf{pid} = A$.*

A note on out-of-order messages. A property not considered in our analysis is *out-of-order* (OOO) messaging (or message-loss resilience [4]). Both Signal and SAID provide this feature by design, allowing intermediate messages that are lost to still be recovered in spite of the key having evolved beyond that point. This implies storing several message or chain keys in memory (until they can be used). There are two consequences to our including OOO messaging. First, this implies that even when having matching conversation, Alice and Bob might have non-identical session identifiers (one might have "holes" in the messages that are received). Second, message keys are now computed and stored beyond a single stage. In our formalization, this changes the type of security that is achieved with respect to the values we present in Sec. 3. However, note that capturing message-loss resilience is conceptually compatible with our model.

Correctness. If π_A^i and π_B^j have matching conversation, then a SCEKE protocol `SCEKE` is *correct* if both conditions hold:

- for each $s = (x, y)$, both instances have identical $\mathsf{mk}^{x,y}$;
- A uses `aSend` to output (π_P^i, C, AD^*) from M and B inputs C for `aReceive` then π_A^i and π_B^j are still matching.

2.2 Adversarial Model

Our adversary is a Probabilistic Poly-Time adversary $\mathcal{A}$, which manipulates honest parties by using *oracles*. Depending on $\mathcal{A}$'s strength (see next section for taxonomy) the attackers may query all, or just some of the oracles presented below.

For reasons that will become apparent when we present our taxonomy, we divide the private keys that parties use during SCEKE sessions into three categories:

Cross-session Keys: keys that (intentionally) repeat in at least two sessions[3]. More formally, a key k is cross-session if there exist distinct instances π_P^i, π_P^j of registered party P, and (potentially) distinct stages $s \in \pi_P^i$ and $s' \in \pi_P^j$ such that $k \in \pi_P^i\mathsf{var}[s]$ and $k \in \pi_P^j\mathsf{var}[s']$. By definition, the identity and registration keys ik_P and sk of P are cross-session. We denote the set of cross-session keys of party P as $P.X\mathsf{sid}$.

Cross-stage Keys: keys that (intentionally) repeat in at least two stages of the same session, but not across sessions. More formally, there exists an instance π_P^i and distinct stages $s \in \pi_P^i$ and $s' \in \pi_P^i$, such that $k \in \pi_P^i\mathsf{var}[s]$ and $k \in \pi_P^i\mathsf{var}[s']$, but $k \notin P.X\mathsf{sid}$. We denote the set of cross-stage keys belonging to instance π_P^i as $\pi_P^i.X\mathsf{stage}$.

Stage-specific Keys: keys occurring in only one stage of one protocol instance π_P^i, *i.e.*, $k \in \pi_P^i.\mathsf{var}[s]$ for some stage s, but $k \notin (P.X\mathsf{sid} \bigcup \pi_P^i.X\mathsf{stage})$. We denote by $\pi_P^i.\mathsf{1stage}$ the set of all stage-specific keys of instance π_P^i.

Oracles. The adversary can register malicious users; compromise users to reveal, respectively, cross-session, cross-stage, and stage-specific private values; and manipulate communication by instantiating new sessions and sending/receiving messages. The adversary's goal will be distinguished from random a target message-key that is freshly and honestly generated. Thus, each instance needs to also store the attribute $\pi_P^i.\mathsf{b}[s]$: a challenge bit randomly chosen for each instance for stage s. If $b = 1$, the output is the real message key, else the output is a random key.

We describe the PCS-game in Sec. 2.4, while a taxonomy of adversarial *types* are introduced in Sec. 2.3. In the game, the adversary may query (a subset of) the following oracles:

$\underline{\mathsf{oUReg}(P)}$: runs `aKeyGen` on party P *i.e.*, $\mathcal{A}$ can register malicious P to an honest $\hat{S}$.

$\underline{\mathsf{oStart}(P, \mathsf{role}, \mathsf{pid}, hon)}$: runs $\Pi_{\mathtt{Start}}$ to create a new instance of an existing honest party with the role role and intended partner pid. The added value hon is a bit, which, if set to 1, runs the protocol with the challenger posing as $\hat{S}$, whereas if $hon = 0$, the protocol is run with the adversary posing as $\hat{S}$.

$\underline{\mathsf{oTest}_b(\pi_P^i, s)}$: for honest parties, valid instances, valid stages, and a computed message-key at stage s, returns that key (if $\pi_P^i.\mathsf{b}[s] = 1$) or a random key of the same length ($\pi_P^i.\mathsf{b}[s] = 0$). This oracle can only be queried once.

$\underline{\mathsf{oSend}(\pi_P^i, s, AD)}$: two modes for this oracle: honest or maliciously-controlled. For $AD = \perp$ (other values are valid), π_P^i generates new key pair using `aRGen` for stage s then it runs `aSend`, and outputs the additional data. Otherwise, the oracle simulates the sending algorithm with adversarially-chosen AD.

$\underline{\mathsf{oReceive}(\pi_P^i, s, AD)}$: oracle in two modes. In honest mode, AD is valid since output by oSend at stage s by π_P^i's partner. For the adversarial mode, AD is always considered correct (*e.g.*, allowing communication hijacking).

$\underline{\mathsf{oReveal.XSid}(P)}$: corrupts P, giving $\mathcal{A}$ access to $P.X\mathsf{sid}$.

$\underline{\mathsf{oReveal.XStage}(\pi_P^i, s)}$: for stage s, it leaks the set of keys $\pi_P^i.X\mathsf{stage} \bigcap \pi_P^i\mathsf{var}[s]$ of cross-stage values.

$\underline{\mathsf{oReveal.1Stage}(\pi_P^i, s)}$: for stage s, it leaks the set $\pi_P^i.\mathsf{1stage} \bigcap \pi_P^i\mathsf{var}[s]$ of stage-specific values.

Like in [7], $\mathcal{A}$ does not have access to the real ciphertext, which is a trivial distinguisher.

2.3 A taxonomy of adversaries

We classify adversaries in terms of: reach; power; and access, as discussed below. Although the security games and winning conditions are mostly equivalent, different adversaries will learn a different subset of values upon compromising a party, and will be allowed different sequences of oracle queries.

Reach. Our model features three types of corruption oracles: oReveal.XSid, oReveal.XStage, and oReveal.1Stage, revealing, respectively, the party's cross-session (long-term) keys, cross-stage keys, and stage-specific keys. Of these, the latter are assumed to be the least protected because as they are the least impactful during key-evolution. We distinguish:

<ins>Local adversaries:</ins> are only allowed access to the oReveal.1Stage oracle;

<ins>Medium adversaries:</ins> may query both oReveal.1Stage and oReveal.XStage, but not oReveal.XSid;

<ins>Global adversaries:</ins> may query all three oracles.

[3]We thus formally exclude collisions in randomness

Power. We distinguish between attackers which extract information from honest participants via their reveal oracles, and stronger adversaries, which extract data and then use it to hijack honest sessions, or for other (evil) purposes. This reasoning leads to a classification between:

Active adversaries: The attacker may use the malicious modes of the oSend and oReceive oracles on the target instance π_P^i, or on the instance it has matching conversation with. We define below one potential strategy of such attackers, namely session hijacking, but active adversaries are not restricted to only it. In short, (successfully) hijacking a session enables the adversary to insert its own key material and increase the interval required before healing (or make the channel unable to heal);

Passive adversaries: These attackers may not use the malicious modes of the sending and receiving oracles on the target instance, nor its partner.

We define the *hijacking* of a session run between π_A^i and its partner π_B^j at some stage $s_h = (x_h, y_h)$ (for which we assume w.l.o.g. that A is the sender) the event that the following conditions hold simultaneously:

1. $\mathcal{A}$ has queried $\mathsf{oReceive}(\pi_B^j, s_h, AD_h)$;
2. AD_h were never output by an $\mathsf{oSend}(\pi_A^i, s_h, \cdot)$ query;
3. there exists a value $v \in AD_h$, but such that $v \notin \pi_A^i.\mathsf{var}[s_h] \cup \pi_B^i.\mathsf{var}[s_h]$.

We call stage s_h *successfully hijacked* if in addition the oReceive query in 1. yielded an output different from $\bot$.

Access. The last criterion in our taxonomy is access. Typically, channel-security is defined with respect to a Person-in-the-Middle attacker. However, some such protocols also feature a centralized entity with more extensive access and thus greater potential to wreak havoc – in our framework, the super-user $\hat{S}$. We divide attackers into two categories:

Insider adversaries: they *are* in fact the super-user. Throughout the game, they receive from the challenger all the private keys and database information amassed by $\hat{S}$.

Outsider adversaries: these attackers do not receive any $\hat{S}$ data. Since additionally $\mathcal{A}$ has no oracle-access to corrupting $\hat{S}$, the latter will remain honest.

Adversarial types. We consider adversaries whose types are a composition of three characteristics, in the order (power, reach, access). The weakest adversary is a passive local outsider. The strongest is an active global insider. All other characteristics being equal, active attackers are stronger than passive ones; also, global attackers are stronger than medium ones, which are in turn stronger than local ones; finally, insiders are stronger than an outsiders.

Nevertheless, intermediate adversaries with more than two varying characteristics are not as easy to compare. This is particularly the case for insider attacks, for which the information obtained by the insider is highly protocol-specific. The same holds for active local adversaries versus passive global ones. In our case, moreover, comparing such adversaries asymptotically is not as interesting as quantifying, for each adversary, the exact healing speed of the scheme. In Fig. 2, we recap the adversary's access to oracle depending on its type.

YYY	LPO	MPO	GPO	LPI	MPI	GPI	LAO	MAO	GAO	LAI	MAI	GAI
1Stage	✓			✓			✓			✓		
XStage		✓			✓			✓			✓	
XSession			✓			✓			✓			✓
Access $\hat{S}$.sk				✓	✓	✓				✓	✓	✓
oReceive	H	H	H	H	H	H	✓	✓	✓	✓	✓	✓
oSend	H	H	H	H	H	H	✓	✓	✓	✓	✓	✓
oUReg	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
oStart	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 2: Available oracles depending on type, labelled reach||power||access. For instance, LAO denotes Local Active Outsider adversary. We omit oTest oracle since all adversaries may query it. H denotes an honest call to the oracle.

2.4 A metric for PCS

The adversary $\mathcal{A}$ plays against a challenger $\mathcal{C}$ in the following security game $\mathsf{Exp}_{\mathsf{SCEKE}}^{PCS}(\lambda, \mathcal{A})$, which is depicted in Fig. 3:

- $\mathcal{C}$ runs `aSetup` and forwards all the public values to $\mathcal{A}$. $\mathcal{C}$ also simulates the registration of all the honest parties.
- $\mathcal{A}$ has access to algorithms `aKeyGen` and `aRGen` and, depending on its type, may adaptively query a subset of these oracles (see also Fig. 2):
  - $\mathsf{oUReg}(P)$ (all attackers);
  - $\mathsf{oStart}(P, \mathsf{role}, \mathsf{pid}, 1)$ (outsider $\mathcal{A}$) and $\mathsf{oStart}(P, \mathsf{role}, \mathsf{pid}, 0)$ for (insiders);
  - $\mathsf{oSend}(\pi_P^i, s, \bot)$ (passive $\mathcal{A}$) and $\mathsf{oSend}(\pi_P^i, s, AD)$ (active $\mathcal{A}$);
  - $\mathsf{oReceive}(\pi_P^i, s, \bot)$ (passive $\mathcal{A}$) and $\mathsf{oReceive}(\pi_P^i, s, AD)$ (active $\mathcal{A}$);
  - $\mathsf{oReveal.XSid}(P)$ (global $\mathcal{A}$);
  - $\mathsf{oReveal.XStage}(\pi_P^i, s)$ (medium $\mathcal{A}$);
  - $\mathsf{oReveal.1Stage}(\pi_P^i, s)$ (local $\mathcal{A}$).
- At some point, $\mathcal{A}$ outputs a party instance $\pi_P^\star$ and a stage $s^\star = (x^\star, y^\star)$. The challenger $\mathcal{C}$ runs $\mathsf{oTest}_b(\pi_P^\star, s^\star)$ and outputs the true $\pi_P^\star.\mathsf{mk}^s$ or a random key.
- The attacker may continue to use its oracles/algorithms, until it outputs a final bit d.

We say that $\mathcal{A}$ wins $\mathsf{Exp}_{\mathsf{SCEKE}}^{PCS}(\lambda, \mathcal{A})$ if and only if $d = \pi_P^\star.\mathsf{b}[s^\star]$, and if the winning conditions below hold. The *advantage* of the adversary is computed as: $|\Pr[\mathcal{A} \text{ wins } \mathsf{Exp}_{\mathsf{SCEKE}}^{PCS}(\lambda, \mathcal{A})] - \frac{1}{2}|$.

Further winning conditions. Adversary $\mathcal{A}$ wins $\mathsf{Exp}_{\mathsf{SCEKE}}^{PCS}(\lambda, \mathcal{A})$ if it guesses the real-or-random bit b for the target message key, and must do so by a non-trivial attack (*e.g.*, not revealing the target message key). Attacks are classified as trivial or non-trivial depending on adversary type. We express them as a conjunction of predicates parametrized by $\mathcal{A}$'s type and resulting PCS security.

```
$\underline{\mathsf{Exp}^{PCS}_{\mathsf{SCEKE}}(\lambda, \mathcal{A})}$
$(\hat{S}.\mathsf{sk}, \hat{S}.\mathsf{pk}, \mathsf{pparam}) \leftarrow \mathcal{C}^{\mathtt{aSetup}(1^\lambda)}$
$(\mathcal{P} = \{P_1, \cdots P_{\mathsf{n}_\mathcal{P}}\}) \leftarrow \mathcal{C}(\lambda, \mathsf{n}_\mathcal{P})$
$(\mathsf{ik}_i, \mathsf{ipk}_i) \leftarrow \mathcal{C}^{\mathtt{aKeyGen}(1^\lambda)} \quad \forall i \in \{1, \cdots, \mathsf{n}_\mathcal{P}\}$
$\mathcal{O}_{\mathsf{type}} \leftarrow \left\{ \begin{array}{l} \mathsf{oUReg}(\cdot), \mathsf{oStart}(\cdot,\cdot,\cdot,\cdot), \mathsf{oReveal}[\mathcal{A}.\mathsf{reach}](\cdot,\cdot), \mathsf{oSend}(\cdot,\cdot,\cdot,\cdot), \\ \mathsf{oReceive}(\cdot,\cdot,\cdot,\cdot),, \mathcal{RO}_1(\cdot), \mathcal{RO}_2(\cdot) \end{array} \right\};$
$(\pi^\star_P, s^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{type}}}(1^\lambda)$
$K \leftarrow \mathsf{oTest}_{b^\star}(\pi^\star_P, s^\star)$
$d \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{type}}}(\lambda, \mathsf{n}_\mathcal{P}, K)$
$\boxed{\mathcal{A} \text{ wins iff. } d = b^\star \text{ and } (\neg\mathsf{oUReg}(P) \vee \neg\mathsf{oUReg}(\pi^i_P.\mathsf{pid})) = \top}$
```

Figure 3: The PCS game $\mathsf{Exp}^{PCS}_{\mathsf{SCEKE}}(\lambda, \mathcal{A})$ between adversary $\mathcal{A}$ and challenger $\mathcal{C}$, parametrized by the security parameter λ and number of *honest* parties $\mathsf{n}_\mathcal{P}$. $\mathcal{A}$ can query a set of oracles $\mathcal{O}_{\mathsf{type}}$, subject to type. We denote by $\mathsf{oReveal}[\mathcal{A}.\mathsf{reach}]$ the precise reveal oracle allowed to $\mathcal{A}$, subject to its reach (local, medium, or global).

Definition 3 ((χ, Υ)-PCS security) *A* SCEKE *protocol is* (χ, Υ)*-PCS-secure against an adversary* $\mathcal{A}$*, for* $\chi, \Upsilon \in \mathbb{N}$ *and* $\mathcal{A}$ *of one of the* 12 *types above if, and only if, assuming* oTest *will be queried for instance* π^i_P*, the last stage for which* $\mathcal{A}$ *queried* oReveal.XStage *or* oReveal.1Stage *for either* π^i_P *or its matching instance is* $s^* = (x^*, y^*)$ *and the following conditions hold:*

- *The adversary has a non-negligible advantage to win the game* $\mathsf{Exp}^{PCS}_{SCEKE}(\lambda, \mathcal{A})$ *when querying* oTest *for* $s_{\mathsf{Test}} = (x_{\mathsf{Test}}, y_{\mathsf{Test}})$ *such that:*
  - *If* $\Upsilon = 0$*,* $x_{\mathsf{Test}} < x^* + \chi$ *and* $y_{\mathsf{Test}} = y^*$*;*
  - *If* $\Upsilon > 0$*,* x_{Test} *is arbitrary and* $y_{\mathsf{Test}} < y^* + \Upsilon$*.*

  If, moreover, the adversary is allowed to query oReveal.XSid*, then* $\mathcal{A}$ *has a non-negligible chance to win for all instances of party P which are not yet instantiated, or have not yet reached stage* $s = (x, y)$ *such that:*
  - *If* $\Upsilon = 0$*, then* $x \geq \chi$ *and* $y \geq 1$*;*
  - *If* $\Upsilon > 0$*, then* $x > 1$ *and* $y \geq \Upsilon$*.*
- *The adversary has a neglibile advantage to win if* oTest *is queried for* s_{Test} *other than those specified in the first bullet point.*

We allow both χ *and* Υ *to take a special value* ∞*, which corresponds to "an arbitrary number of stages" obtained through horizontal and respectively through vertical evolution.*

3 Case Studies

We apply our metric to 3 use cases: the PKI-based messaging protocol Signal, the Identity-Based messaging protocol SAID, and the suite of mobile 5G Handover protocols. Although seemingly very different, they all can be modelled as SCEKE schemes, which shows the generality of our framework.

	Outsider	Reach	Signal	SAID	*5G-SCEKE*	*5G-SCEKE*$^+$
Access	Passive	Global	$(\infty,2)$	$(\infty,2)$	(∞,∞)	$(1,0)$
		Medium	$(\infty,2)$	$(\infty,2)$	(∞,∞)	$(1,0)$
		Local	$(\infty,1)$	$(1,0)$	$(\infty,1)$	$(1,0)$
	Active	Global	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)
		Medium	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)
		Local	$(\infty,1)$	$(1,0)$	$(\infty,1)$	$(1,0)$

	Insider	Reach	Signal	SAID	*5G-SCEKE*	*5G-SCEKE*$^+$
Access	Passive	Global	$(\infty,2)$	$(\infty,2)$	(∞,∞)	(∞,∞)
		Medium	$(\infty,2)$	$(\infty,2)$	(∞,∞)	(∞,∞)
		Local	$(\infty,1)$	$(\infty,1)$	(∞,∞)	(∞,∞)
	Active	Global	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)
		Medium	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)
		Local	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)

Figure 4: Results for our metric on PCS-security for Signal, SAID, 5G handover and its variant denoted Ext-*5G-SCEKE*.

3.1 Signal as a SCEKE protocol

Signal is a natural instantiation of SCEKE protocols. Like most asynchronous-messaging schemes, Signal conversations are turn-based, between speakers who need not be online simultaneously. Each message corresponds to one stage, *i.e.*, each message-key is used only once. New keys are generated in two different ways: when the person who was speaking sends a new message, we have a horizontal evolution; when the speaker changes, we have a vertical evolution.

Signal also features a natural super-user: a centralized credential server storing user public keys. Our SCEKE framework includes security with respect to such powerful insiders, an aspect often overlooked by prior work [13].

We first compare our model with the one by Cohn-Gordon *et al.* [13]; then we model Signal as a SCEKE protocol. Our extended version [8] provides detailed description. Then we quantify PCS-security with respect to all the adversaries in Sec. 2.3. The security proofs are given in Appendix A.

Comparing security models. Our framework can be seen, in many ways, as a generalization of Cohn-Gordon *et al.*'s Signal-specific security model [13]. They described a real-or-random key-indistinguishability experiment akin to ours, for which the Person-in-the-Middle adversary $\mathcal{A}$ can test stages freely in order to distinguish their message-keys from random. $\mathcal{A}$ wins assuming that it guesses correctly and that a given freshness predicate holds.

We begin by stating that the adversary described by [13] is a passive outsider: they rule out adversarial interventions within the target session, and do not consider security with respect to the super-user. Finally, the oracles they consider are slightly different from ours, as we describe below.

From Fig. 5, we can infer that:

$\mathsf{oReveal.1Stage} \implies \mathsf{RevSessKey} \wedge \mathsf{RevRand} \wedge \mathsf{RevStateMiddle}$
$\mathsf{oReveal.XStage} \implies \mathsf{RevRand} \wedge \mathsf{RevStateInit}$
$\mathsf{oReveal.XSid} \implies \mathsf{RevLongTermKey} \wedge \mathsf{RevRand}$

Thus the adversaries captured in [13] can adopt more fine-grained strategies than ours. For instance, in our model, if the adversary wants a particular cross-stage key, it essentially will

	ms	ephk	ck	mk	rk	rchk	ik	prek
oReveal.1Stage	✓	✓	✓	✓				
oReveal.XStage	✓	✓	✓	✓	✓	✓		
oReveal.XSid	✓	✓	✓	✓	✓	✓	✓	✓
RevSessKey				✓				
RevLongTermKey							✓	
RevMedTermKey								✓
RevRand		✓				✓		
RevStateInit					✓			
RevStateMiddle			✓					

Figure 5: Revealed keys per oracle queries: ✓s indicate revealed keys. The 3 upper rows list oracles in our model, while the bottom ones are oracles from [13]. Notice that for Signal, we split oracle RevState into RevStateInit (which can be used only at the beginning of a stage-chain) and RevStateMiddle (for queries inside a chain *i.e.*, $x > 1$).

receive *all* such keys. As a consequence, we lose the ability to rank, say, cross-stage keys in terms of how dangerous they are to healing. Yet, (instantiations of) the predicates described above are in fact also found amongst the winning conditions of [13], signifying that the same

Yet, in reality (as described in the proofs), the winning predicates of [13] imply that the adversary does not essentially benefit from the additional freedom given by those fine-grained queries. Thus, while our two frameworks are syntactically incomparable, they are akin in spirit. In addition, our model allows us to account for additional adversary types, including active adversaries and insiders.

Description. Let $\mathcal{P}$ be a set of honest users (with unique identities). Our super-user $\hat{S}$ is a centralized PKI server.

<u>SETUP</u>. During the global setup of the protocol, the super-user $\hat{S}$ chooses a DH-based signature algorithm, hash functions and KDFs, and a secure-channel establishment protocol required at registration (such as TLS)[4]. Then, $\hat{S}$ generates keys required for its authentication in the secure-channel establishment, notably $(\hat{S}.\mathsf{sk}, \hat{S}.\mathsf{pk})$. We also assume that each user in $\mathcal{P}$ has keys allowing it to register over a secure channel to $\hat{S}$, but make no assumptions as to their structure; we merely require that they provide secure authentication. All the algorithms and $\hat{S}.\mathsf{pk}$ are part of the public system parameters.

<u>KEY GENERATION</u>. During key-generation, each party generates signature identity keys $(\mathsf{ik}_P, \mathsf{ipk}_P)$ for the signature scheme chosen at setup.

<u>USER REGISTRATION</u>. Over an AKE-secure channel between $\hat{S}$ and each user P, the latter registers a key-bundle consisting of: a long-term identity key ipk_P, a medium-term key prepk_P signed with ik_P, and optional ephemeral *public* keys ephpk_P. Both ipk_P and prepk_P are used across multiple sessions, whereas each ephemeral public key ephpk_P is

[4]In other words, we assume that whenever they upload keys to the server, parties do so over a mutually-authenticated secure channel (with standard AKE security).

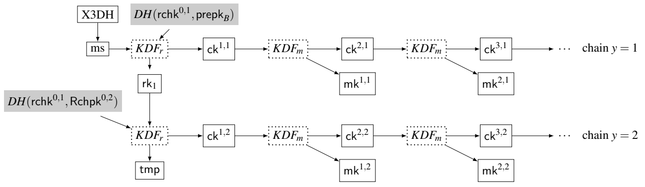

Figure 6: Signal's key schedule, in which vertical evolutions are boxed in grey. Horizontal evolutions are along chains represented horizontally.

only used in one session and then removed from the server. Subsequent calls to this algorithm allow users to update key material they have previously submitted. We stress that the server is never given the user's private keys.

<u>INSTANCE INITIALISATION</u>. Alice (the initiator) begins a session with Bob by querying the semi-trusted server, over an authenticated channel, for Bob's credentials, which allows Alice to establish the master secret ms.

The master secret will yield an intermediate *root key* rk_1 and the first *chain key* $\mathsf{ck}^{1,1}$; the latter will be input to a key-derivation function (KDF) in order to output a new key $\mathsf{ck}^{2,1}$ and the first *message key* $\mathsf{mk}^{1,1}$, which will be used to authenticate and encrypt Alice's first message to Bob, corresponding to stage $(1,1)$ of the session.

<u>SENDING AND RECEIVING</u>. For the remainder of the session, Alice and Bob exchange encrypted messages. On stages with odd y, Alice is the sender and Bob is the receiver, while on stages with even y, it is the other way around. In each stage, corresponding to a single encrypted message, the included metadata allows that message's receiver to make his keys evolve, either horizontally (it receives and decrypts a new message) or vertically (the receiver decides to start talking).

Fig. 6 gives this key-derivation process.

The associated metadata at stage (x,y) consists of the identities of the two speakers, a ratchet public key Rchpk^y (usable for vertical evolution), and the index x of the message. Exceptionally, for messages sent at stages $(\cdot, 1)$, the metadata must also include the public key Epk_A corresponding to Alice's private key ek_A used during session initialisation. This metadata is sent as Associated Authenticated Data (AAD) within each AEAD-encrypted ciphertext[5]. Thus, this data passes in clear, but is authenticated as part of the ciphertext.

The PCS-security of Signal. We begin the analysis by splitting the key material used in Signal into stage-specific, cross-stage, and cross-session keys.

The keys used in Signal for a single stage only are: the message[6] keys $\mathsf{mk}^{x,y}$, the chain keys $\mathsf{ck}^{x,y}$, and also a particular key used only at stage $(1,1)$, namely ephk.

On the other hand, private ratchet keys $\mathsf{rchk}^{x,y}$ and the keys

[5]AEAD stands for Authenticated Encryption with Associated Data.

[6]We explicitly do not consider the fact that in Signal keys can actually be precomputed in the case of out-of-order arrivals, since this is not the most frequent way in which the protocol is used.

Keys	Signal	SAID	*5G-SCEKE* or *5G-SCEKE*$^+$
Cross-Session	ik,prek	ik,ID.sk, IBS.sk	K
Cross-Stage	rk,rchk, *tmp*	ms, rk,rchk	$\mathsf{K}_{AMF} = \mathsf{rk}$
Single-Stage	ephk, ms,mk, ck	mk,ck, *r*	rchk =v , $\mathsf{K}_{AS} =$ mk, $\mathsf{K}_{gNB} = $ ck

Table 1: Taxonomy on keys used in Signal, SAID and our 5G handover procedures model.

used at the roots of each chain(denoted rk_y for odd y and tmp^y for even y) are stored throughout the existence of the chain, until the next vertical evolution. In other words, they are cross-stage keys. The key material is summarized in Table 1.

Theorem 1 *Consider the Signal protocol modelled as a SCEKE scheme, as presented above. The following results hold in the random oracle model (by replacing the KDFs with random oracles), under the Gap Diffie-Hellman assumption, and assuming the AKE security of the channels established between honest users and an honest $\hat{S}$:*

- *Signal is $(\infty, 1)$-PCS secure against: local outsiders (passive and active), local passive insiders;*
- *Signal is $(\infty, 2)$-PCS secure against: medium passive adversaries (outsiders and insiders), and global passive attackers (outsiders and insiders);*
- *For all other adversaries, Signal is (∞, ∞)-PCS secure.*

Note that the results are also systematized in Fig. 4. The proofs of this theorem consist of two types of statements: first, we need to show an attack for the stages that are vulnerable to the attacker, then we need to prove that beyond those stages, security holds. The second parts of the proofs can be found in Appendix A.1, but we briefly indicate the attacks providing the first part of the proofs below.

LOCAL PASSIVE OUTSIDER Here the security loss is a result of the symmetric ratchets: once $\mathsf{ck}^{x,y}$ is compromised, $\mathcal{A}$ learns all the chain and message keys derived symmetrically from it. The ratchet key rchk^y is not amongst the data revealed through oReveal.1Stage. When it is used in input at stage (1,y+1), $\mathcal{A}$ can no longer compute keys derived from this key.

MEDIUM PASSIVE OUTSIDER As opposed to the previous case, the attacker can now query oReveal.XStage and learn ratchet keys rchk, and root keys. Knowledge of the ratchet key rchk^y allows $\mathcal{A}$ to compute $\mathsf{DH}^{0,y+1} = DH(\mathsf{Rchpk}^y, \mathsf{Rchpk}^{y+1})$ at the beginning of chain $y+1$ and derive all the keys in chain $y+1$ (hence implying $(\infty, 2)$-PCS security). Fortunately, this stops at stage $(1, y+2)$, since $\mathcal{A}$ cannot use rchk^y to compute $\mathsf{DH}^{0,y+2}$, thus giving the PCS bound.

GLOBAL PASSIVE OUTSIDER The adversary's access to oReveal.XSid provides user identity keys and pre-keys. However, these values cannot help a passive adversary beyond learning the master secret ms (via oReveal.1Stage). This essentially reduces a global passive outside to a medium passive adversary.

LOCAL ACTIVE OUTSIDERS In this weakest form of active outsider attacks, the attacker can still actively *use* the information captured through corruption, in addition to learning it by compromising either endpoint. Unfortunately, this is not helpful, since in order to go beyond the $(\infty, 1)$ bound provided in the local passive outsider case, $\mathcal{A}$ would require knowing the chain's current root key. Although this is a Denial of Service (DoS) attack, it will not affect PCS security.

OTHER ACTIVE OUTSIDERS The attacker has access to oReveal.XStage, and so to the root key it was missing in the previous cases. As a result, the attacker can use its active capacity to learn the message and chain keys of some stage $(1, y)$ and then *use them* to inject its own ratcheting information, towards the receiver at chain y. Then, by using the root key (via oReveal.XStage) it keeps up with all future ratchets. This compromises all the future keys in these sessions, yielding an (∞, ∞)-PCS security.

INSIDER PASSIVE ATTACKS The knowledge of the super-user's private key $\hat{S}$.sk will not help the adversary beyond an outsider adversary's capacity. This is the situation that corresponds to an honest-but-curious server – for which Cohn-Gordon *et al.* considered (and proved) the security we also explained for the outsider case. This explains why we have the same bounds for the insider and outsider passive adversaries.

INSIDER ACTIVE ATTACKS At the opposite end of the scale are active insider attacks, which basically capture a fully malicious centralized server. At user registration, the malicious super-user behaves as normal. However, at session setup, when Alice wants to talk to Bob, $\hat{S}$ forwards a key-bundle of its own making, to which it has the corresponding private keys. The attacker then does the same when Bob asks for Alice's credentials (forwarding keys from the same bundles), thus ensuring that it can run a Person-in-the-Middle attack between the two users. This type of attack requires no reveal queries on any of the user key material – hence, Signal provides (∞, ∞)-PCS security (no healing at all) for all insider active attackers.

Signal with acknowledgements. More recent implementations of Signal have slightly evolved from the core protocol we described in this paper, and have added an acknowledgement, which essentially reduces message-chain length to 1. In addition, root and ratchet keys become stage-local keys, thus augmenting security against local adversaries to $(\infty, 2)$.

Signal with two-factor authentication. A way to reduce the impact of insider attacks is to have users verify the identity keys of other users prior to instantiating sessions with them – a type of two-factor authentication. However, such verifications are not without dangers, as described in recent literature [16].

3.2 SAID as a SCEKE protocol

Introduced in 2019, SAID's main aim is to strengthen authentication in messaging protocols [7]. The protocol is proposed in the identity-based (IB) setting, requires a Key-Derivation Center to replace Signal's credential server, and makes substantial modifications to key-evolution. We briefly review SAID here, and include more details in our extended version [8]

Protocol description. Our description below follows that of [7], but expands upon the user-registration part. We notably consider additional keypairs for the KDC and protocol participants – which will allow them to establish the secure channel they require at registration. Although these keys do not feature in SAID, Blazy *et al.* [7] do suppose that a mutually-secure channel exists during that process.

SETUP. SAID relies on an identity-based signature scheme $\mathsf{IBSig} = (\mathtt{aIBS.Setup}, \mathtt{aIBS.Extract}, \mathtt{aIBS.Sign}, \mathtt{aIBS.Vfy})$ and a type-3 pairing e. At system setup the KDC generates global public and private parameters. It must notably generate global setup values $(\mathsf{IBS.msk}, \mathsf{IBS.mpk})$ for the IB signature scheme and parameters $\mathsf{ID.msk} \xleftarrow{\$} \mathbb{Z}_p$ (private) and $\mathsf{ID.mpk} = g_2^{\mathsf{ID.msk}} \in \mathbb{G}_2$ (public) for embedding identities into private identity keys. $\hat{S}$ generates a key-pair that enables authentication in the AKE protocol of its choice (*e.g.*, TLS 1.3), denoting them $(\hat{S}.\mathsf{sk}, \hat{S}.\mathsf{pk})$, then appends $\mathsf{ID.msk}$ and $\mathsf{IBS.msk}$ to $\hat{S}.\mathsf{sk}$.

KEY GENERATION. This step proceeds as in [7], but we additionally have parties register some non-IB keypairs $(\mathsf{ik}, \mathsf{ipk})$, usable during registration.

REGISTRATION. Users P register over a secure channel established with the KDC (P uses its $(\mathsf{ik}, \mathsf{ipk})$ tuple and KDC, $(\hat{S}.\mathsf{sk}, \hat{S}.\mathsf{pk})$). Over this channel, P sends her identity P (*e.g.*, a phone number, email address, etc.), to the KDC. The KDC returns the user's secret signing key $\mathsf{IBS.sk}_P \leftarrow \mathtt{aIBS.Extract}(\mathsf{IBS.ppar}, \mathsf{IBS.msk}, P)$ and secret identification key $\mathsf{ID.sk}_P = H_2(P)^{\mathsf{ID.msk}}$. Thus, the KDC knows all the users' private keys.

INSTANCE INITIALISATION. In SAID, instance-initialisation requires no user-KDC interaction (thus we deem $\hat{S}$'s contribution void). Initiator Alice will choose randomness r and compute $\mathsf{ms}_{AB} = e(H(B), \mathsf{ID.mpk})^r$: in other words, Alice embeds the Bob's identity into the master secret. Alice also generates a random *tag*: $\mathsf{tag}^{1,1}$, and uses it and the master secret to derive the root key rk_1 and the first chain key $\mathsf{ck}^{1,1}$. The use of fresh tags is specific to SAID and ensures that keys are unlikely to repeat. A KDF is used to derive the chain key $\mathsf{ck}^{2,1}$ and first message key $\mathsf{mk}^{1,1}$ from ms_{AB} and $\mathsf{ck}^{1,1}$.

Unlike Signal, SAID uses the master secret ms_{AB} at every stage; thus Bob has to regularly prove knowledge of his private identity-key, and Alice, of the secret r signed with her IB signing key. In [7] all parties store values ik_P and master secrets ms_{P*} of started sessions, and ms_{*P} of responded sessions in a *trusted execution environment* – which we abstract.

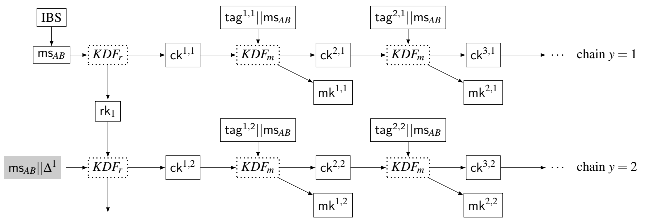

Figure 7: The key schedule of SAID.

SENDING AND RECEIVING. Stages and keys evolve in SAID similarly to Signal:

- **Symmetric ratcheting:** To go from stage (x, y) to $(x+1, y)$, the current speaker generates a new tag $\mathsf{tag}^{x+1,y}$ to be input with chain key $\mathsf{ck}^{x+1,y}$, in order to output $\mathsf{ck}^{x+2,y}$ and $\mathsf{mk}^{x+1,y}$. The two precise substeps are detailed in [8]
- **Asymmetric ratcheting:** When speakers change, the key material is freshened up with Diffie-Hellman randomness: on input the master secret, a value $\Delta^{y-1} = DH(\mathsf{Rchpk}^{0,y-1}, \mathsf{Rchpk}^{0,y})$, and the root key rk_y, a KDF outputs rk_{y+1} and $\mathsf{ck}^{1,y+1}$. The chain key, master secret, and a fresh tag $\mathsf{tag}^{1,y+1}$ are used to obtain the chain's first message key $\mathsf{mk}^{1,y+1}$.

The receiver gets the public key material allowing it to ratchet correctly as authenticated metadata. For the first message chain, Alice sends the following AAD: the public value $h = g_2^r$ corresponding to the secret r that the initiator used to compute the master secret; the stage's horizontal index x; a fresh public ratchet key $\mathsf{Rchpk}^1 = g_1^{\mathsf{rchk}^1}$; the tag of the current message; the user identities; and a signature over everything except the tag: $\sigma \leftarrow \mathtt{aIBS.Sign}(\mathsf{IBS.ppar}, \mathsf{IBS.sk}_A, \{\mathsf{meta}_1, h\})$.

For all the messages in chain $y = 2$, we have the same metadata as before, but we no longer need to send h. Starting from $y = 3$, we need to add the number N_{y-2} of messages sent in the previous sending chain (*i.e.*, chain $y-2$). We depict the key-schedule of SAID in Fig. 7.

Comparing security models. Our framework follows closely the model by Blazy *et al.* [7], which describes a real-or-random key-indistinguishability experiment for identity-based secure messaging. Their adversaries are either passive or active outsiders in our taxonomy. The model of [7] has several features identical to ours: a global setup, malicious-user registration procedures, sending, and receiving oracles. Since new-session instantiation is not interactive for SAID, our model boils down to Blazy *et al.*'s on this account.

However, [7] gives different leakage possibilities to its adversaries than we do, through three specific oracles (presented in the lower half of Fig. 8). The first is corruption, which yields our cross-session keys, but also all the master secret values of all ongoing sessions. $\mathcal{A}$ can also reveal a subset of cross- and single-stage keys (specified by name); by contrast, our framework only splits access by key-type (*e.g.*, querying $\mathsf{oReveal.XStage}$ yields all cross-stage keys together).

	r	mk	ck	tag	ms	rk	rchk	ik	ID.sk	IBS.sk
oReveal.1Stage	✓	✓	✓	✓						
oReveal.XStage					✓	✓	✓			
oReveal.XSid								✓	✓	✓
oCorrupt					✓			✓	✓	✓
oReveal		✓	✓	✓		✓	✓			
oHSM					BB			BB	BB	BB

Figure 8: Comparison of leakage oracles for SAID.

Finally, [7] allows $\mathcal{A}$ black-box access to a long-term value: we denote this in Fig. 8 by a "BB" annotation. We describe in detail our classification of keys as stage-local, cross-stage, and cross-session in Table 1 and in the following paragraph.

Although oReveal provides $\mathcal{A}$ more fine-grained access to the local and cross-stage keys (as it can reveal them individually), the SAID protocol proofs make no use of this particular granularity: in other words, security relies on the fact that the adversary is never given access to the master secret (obtained in [7] by a oCorrupt query) simultaneously with the chain or root key allowing $\mathcal{A}$ to compute a target message key.

The PCS-security of SAID. As shown in Table 1, we classify the private-key material of SAID as follows:

- **Stage-specific keys:** These include, per stage: the chain and message keys at that stage, its tag, and the randomness r used only once, at the beginning of the protocol;
- **Cross-stage keys:** Apart from the root and ratchet keys, cross-stage keys now include the master secret, which is input at every stage of the protocol;
- **Cross-session keys:** These include: initial private key ik, identity-based signature IBS.sk and identity keys ID.sk.

We state the following theorem, for which we provide the constructive proofs (PCS security) in Appendix A.2 and concrete attacks (PCS metric) below.

Theorem 2 *Consider the SAID protocol modelled as a* SCEKE *scheme. The following results hold in the random oracle model (by replacing the KDFs and hash functions with random oracles), under the Bilinear Computational Diffie-Hellman assumption, and assuming the EUF-CMA security of the IB-signature scheme* IBSig *and the AKE security of the channels established between honest users and an honest* $\hat{S}$ *at registration:*

- *SAID is* $(1,0)$*-PCS secure against local outsiders (passive and active);*
- *SAID is* $(\infty,1)$*-PCS secure against local passive insiders;*
- *SAID is* $(\infty,2)$*-PCS secure against: medium passive adversaries (outsiders and insiders), and global passive attackers (outsiders and insiders);*
- *For other adversary types, SAID is* (∞,∞)*-PCS secure.*

<u>LOCAL OUTSIDERS</u> For both passive and active outsiders, the SAID PCS bound is $(1,0)$-PCS-secure, which is actually optimal in our framework. The main reason this holds is that the master secret (a cross-stage value) is required for each evolution; hence, an attacker can only at most learn the current message key, but no other.

<u>LOCAL PASSIVE INSIDER</u> Note that, according to our game, insiders might *know* a long-term secret for a user, but they will not *use them*. With its revelation oracle, $\mathcal{A}$ is able to learn the master secret; however, since $\mathcal{A}$ cannot learn root, nor ratcheting keys, it cannot ratchet past a single message-chain.

<u>OTHER PASSIVE ADVERSARIES</u> For global and medium passive insider and outsider adversaries the PCS security limitations are given by the fact that $\mathcal{A}$ *can* learn a ratchet key $\mathsf{rchk}^{x,y}$ and the master secret, but, on the other hand, it is a passive attacker and can thus not *use* that ratchet key for longer than two chains. Moreover, even *passive* knowledge of $\hat{S}$.sk is not helpful. So, SAID is $(\infty,2)$*-PCS secure*.

<u>OTHER ACTIVE OUTSIDERS</u> Knowledge of the master secret is fundamental in SAID. Given this information, an active attacker can hijack the session by including fresh asymmetric ratcheting elements once the corruption has been done. Hence, as in Signal, the protocol never heals ((∞,∞)-PCS security).

<u>ACTIVE INSIDERS</u> We recall that the master secret keys used by the KDC at setup will now be part of the adversary's knowledge, as well as the database of entries containing identities and private keys. This allows the adversary to learn the private keys, both for signatures and their identity keys. This enables the the active, malicious KDC to impersonate Alice towards Bob and Bob towards Alice, thus endangering all their future keys ((∞,∞)-PCS security).

3.3 5G AKE Procedures as A SCEKE Protocol

We showcase the flexibility of our SCEKE framework by modelling a suite of secure-channel establishment protocols in 5th Generation Mobile Networks (5G) as a single SCEKE scheme. The latter will include the well-known Authentication and Key Agreement (AKA) protocol executed during Registration procedure (Reg), and a series of procedures called *handovers* [2,3]. Unlike the two-party SAID and Signal protocols, the suite of 5G protocols we target are run between many entities – we are, nevertheless, able to model them as a two-party SCEKE, with horizontal and vertical key-evolutions.

We briefly describe the 5G context and handovers (a longer description is in [8]). Then we model these procedures as a SCEKE scheme and analyze their PCS security.

The 5G handover protocols. In 5G networks, mobile users (*User Equipment (UE)*) can subscribe to – and receive service from – an operator, whose back-end infrastructure is called the *core network*. The subscribers and the core will share a number of long-term cryptographic secrets denoted succinctly as K. At any point, a user can be given mobile service through a radio "base-station" denoted Next Generation NodeB (gNB), which communicates in parallel with the UE's core network. We assume existence of mutually-authenticated gNB-to-gNB

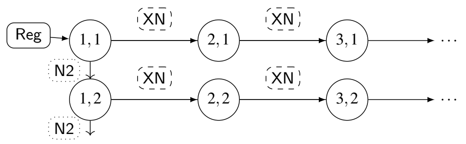

Figure 9: General model of 5G handover procedures.

and gNB-to-core secure channels, and focus on the security of UE-to-gNB and UE-to-Core channels.

Actual mobile/application messages, called *access-stratum messages*, transit between the UEs and a gNB. During initial Registration phase (Reg), these two entities establish initial key material (via AKA protocol). Access-stratum messages are secured with an *access-stratum key (*K_{AS}*)*, which is derived from an intermediate key called k_{gNB} (Signal's chain keys). The latter can be computed through a re-use of the registration procedure, or they can be obtained through evolution via handovers, as we explain next.

Handovers are required when the user connects to a new gNB (*e.g.*, because it physically moves out of reach). At this point, a *handover* is initiated, permitting the evolution of k_{gNB}, from the *source node (s-gNB)* shared by the UE and its old gNB, to a *tgNB*, shared by the UE and its new serving node. There are two handover procedures in 5G: XN handover procedure (XN) denoted here XN, and N2 handover procedure (N2) denoted N2.

The handover procedures. In most cases of handovers, the XN protocol is run. The s-gNB unilaterally evolves its key k_{gNB} into a new key k_{gNB}, which s-gNB securely sends to the target target node (t-gNB). The UE will make its key k_{gNB} evolve by receiving metadata from (the core and) the s-gNB over their secure channel (we call this *horizontal evolution*).

In N2, the *core network* computes the new k_{gNB} for t-gNB by refreshing a larger part of the key-schedule: a vertical evolution. The highest-level key in the 5G key-schedule that can be refreshed by N2 is denoted k_{AMF}. The k_{gNB} keys are lower than k_{AMF}.

The *5G-SCEKE* protocol. We consider a SCEKE protocol that is the composition of the AKA/Reg protocol, which provides some initial key material to the user and network, and multiple, sequential runs of various handover procedures which make that key material evolve. The resulting protocol is denoted *5G-SCEKE*. Each stage corresponds to the protocol establishing, then using, a new k_{gNB}.

Our framework only supports two-party protocols. We thus *compress* the set of all gNB nodes and the core network into a single entity, representing the responder, Bob[7]. The initiator is Alice (the *UE*). The super-user is a key-escrow entity, associating initial key-material to sessions (abstracting AKA).

Fig. 9 presents the key evolution in *5G-SCEKE*, as a

[7]This works in our framework because we require that the endpoints to the target session to be honest – if corrupted. However, note that in some cases in the real world, parts of "Bob" might be malicious.

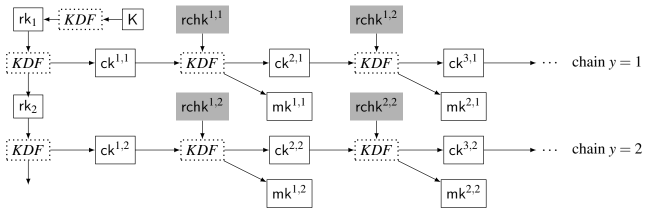

Figure 10: Generic key management for *5G-SCEKE*. The values in grey are modifications only for *5G-SCEKE*$^+$.

SCEKE protocol. Following the registration phase, Alice can horizontally evolve keys by using the XN procedure. When Bob wishes to respond, it runs the procedure N2 to evolve the stage vertically. Thus, in the *5G-SCEKE* protocol the roles are asymmetric: only Alice evolves stage-keys horizontally, and only Bob evolves them vertically.

5G-SCEKE instantiates the initial steps of SCEKE with.
<u>SETUP.</u> The super-user chooses system parameters and generates $\hat{S}.\mathsf{sk}, \hat{S}.\mathsf{pk}$ for secure-channel establishment; <u>KEY GENERATION.</u> We assume that parties create some artificial keys ik, ipk (non-existent in the true 5G context, but needed here to abstract the complexity of AKA); <u>USER REGISTRATION.</u> During user registration, each party P establishes a mutually-authenticated secure channel with $\hat{S}$ and sends a registration request. Then $\hat{S}$ generates one secret K_{PQ} for each Q in its database, but does not send them to P. It updates its database with entry indexed P, with tuples (Q, K_{PQ}) for each existing user Q.
<u>INSTANCE INITIALISATION.</u> Our session instances span the entire duration of *5G-SCEKE*. When Alice initiates a session with Bob, she requests the key $\mathsf{K}A, B$ from $\hat{S}$ over a new secure channel, then uses $\mathsf{K}A, B$ as a master secret and derives, via a KDF, a root key (in practice, K_{AMF}). A KDF computation later, the endpoints obtain the first *chain key* $\mathsf{ck}^{1,1}$ (namely K_{gNB}) and a new root key rk_2. The latter yields a new chain key $\mathsf{ck}^{2,1}$ and a *message key* $\mathsf{mk}^{1,1}$ (notably K_{AS}).

Any of Alice's messages, carrying her identity as metadata, will be sufficient for Bob to initialise a session with her.
<u>SENDING AND RECEIVING.</u> Messages are sent securely in stages, encrypted with the message keys which evolve.

- **<u>Horizontal evolution:</u>** When the initiator wants to send a new message (s goes from (x,y) to $(x+1,y)$), the chain key $\mathsf{ck}^{x+1,y}$ is fed into a KDF to get $\mathsf{ck}^{x+2,y}$ and $\mathsf{mk}^{x+1,y}$.
- **<u>Vertical evolution:</u>** When the responder sends a message (s goes from (x,y) to $(1,y+1)$), a new k_{AMF} is generated from rk_2 and fed into a KDF in order to derive $\mathsf{ck}^{1,y+1}$.

Note that the way to model *5G-SCEKE* as a SCEKE protocol is not unique, and as such, different results could be obtained for different variations of the protocol. Indeed, this is an advantage of our framework, as it allows us to compare those different approaches towards modelling 5G handovers.

The PCS-security of *5G-SCEKE*. We divide key material input into the key-schedule of *5G-SCEKE* viewed as SCEKE:

- **Stage-specific keys:** These include chain keys, associated to $\mathsf{K}^{,}_{gNB}$ and message keys, corresponding to K_{AS}.
- **Cross-stage keys:** The root key, corresponding to K_{AMF}, computed at the beginning of each chain and stored for next vertical evolution.
- **Cross-session keys:** The pre-computed key K shared between each two parties. Each registration procedure corresponds to new instances of all the aforesaid keys, where K is used again.

As we describe in the extended version [8] (and from Fig. 4), the *5G-SCEKE* protocol only provides healing with respect to local outsiders as it lacks any kind of unpredictable freshness. This idea lies at the core of the following improvement that we propose to *5G-SCEKE*.

5G-SCEKE^{+}**: Our Improved-PCS *5G-SCEKE*.** We propose a simple, yet effective, modification of *5G-SCEKE* to enhance the latter's PCS-security, and denote the resulting protocol by 5G-SCEKE^{+}. Notably, we will add freshness into each horizontal evolution, thus limiting the attacker's power. These added values can be viewed boxed in yellow in Fig. 10.

Concretely, we change XN into XN^{+}, a scheme in which s-gNB does not compute the k_{gNB} key for t-gNB. Instead, the latter contributes a locally-generated private value called rchk to k_{gNB} (see the yellow boxes on Fig. 10). Then, t-gNB sends rchk over its secure channel to the core, which in turn forwards it to UE, encrypted with k_{SEAF} (i.e., the key on top of the k_{AMF} in the key-hierarchy in 5G [2, 3]). Now, the UE can also compute the new k_{gNB}. As the sending of rchk can be done on existing XN messages, our modification is minimal.

The key-material in *5G-SCEKE*$^{+}$ is the same as for *5G-SCEKE* except that we add the single-stage keys $\mathsf{rchk}^{x,y}$.

The analysis of *5G-SCEKE*$^{+}$. The following theorem holds for the *5G-SCEKE*$^{+}$ protocol.

Theorem 3 *Consider the* 5G-SCEKE^{+} *protocol as presented above. The following results hold in the random oracle model (by replacing the KDFs with random oracles)*

- 5G-SCEKE^{+} *is* $(1,0)$*-PCS secure against local active outsiders and passive outsiders;*
- *For all other adversary types,* 5G-SCEKE^{+} *is* (∞,∞)*-PCS secure.*

Formal proofs are given in the extended version [8].

Interpreting our *5G-SCEKE*$^{+}$ results. Unlike Signal and SAID for which the rules of evolution are fixed and immutable, 5G handovers can be used in many different *configurations*, and the way users move within the 2D-grid of signal-providing "towers" (i.e., gNBs) impacts their healing interval. Recall that two protocols are used for evolution: XN (or `XN`$^{+}$) – providing horizontal evolution, or N2 – providing vertical evolution. For instance, suppose a gNB which we denote as "Tower1" is configured to only use XN, while some "Tower2" uses only N2. Finally, some "Tower3" can be configured to use the two according to some algorithm: *e.g.*, first time XN, and then N2. If Alice comes across Tower1, then Tower3, then Tower2, she will horizontally evolve twice, then vertically once, while if she goes via Tower1, then Tower2, then Tower3, she will evolve: first horizontally, then vertically, then horizontally again.

The takeaway in this case is two-fold: first, while our results are generic, they can translate to different compromise windows for different configurations and topologies; second, configuring gNBs to use the N2 protocol often and the XN protocol seldomly is the best way of improving the actual healing provided via 5G handovers.

4 Discussion and Conclusion

This paper presents a framework for comparing the post-compromise security achieved by secure-channel establishment protocols featuring key-evolution. Our taxonomy of adversaries includes known adversaries in the literature, but also imagines other type of attacks. The goal of our security definition is not only to prove that key-evolution provides healing, but also to quantify how fast protocols heal. We showcase our framework by applying it to the Signal, SAID, and a composition of AKA and 5G handover protocols. Finally, we also propose a small modification to the latter protocol, which radically improves its healing speed.

Our results (see Fig. 4) indicate that optimal security (*i.e.*, (1,0)-PCS security) is achieved by SAID against local passive outsiders, as well as our improvement of 5G handovers, namely *5G-SCEKE*$^{+}$, against all passive outsiders. An interesting takeaway is the benefit, in *5G-SCEKE*$^{+}$, of using fresh, stage-specific, shared private randomness in the key-derivation process, the unpredictability of which allows us to gain stronger security than SAID for medium and global passive adversaries. However, this security comes at the expense of using shared randomness, which requires secondary secure channels.

We also indicate the benefits of the persistent authentication used in SAID to combat active session-hijacking attacks. Although the use of identifying information into the key computation can be privacy-intrusive (especially if signatures are used), it is able to provide eventual (and even speedy) healing against powerful attackers, otherwise capable of rendering a secure channel unhealable.

Through their reliance on both long-term keys and fresh asymmetric ratchets, Signal and SAID obtain better security against passive insiders than *5G-SCEKE*$^{+}$.

Finally, note that although active insider security is difficult to attain, it is a worthwhile goal. A takeaway of our work is that it is difficult, but essential to design protocols in which users are able to bypass the ability of superusers to create unobservable PitM attacks (for instance, one could consider two-factor authentication of the communication partner).

Our results, while insightful and strong, come with some disadvantages. We only model two-party protocols, and thus cannot analyze multi-user messaging like ART or MLS; yet as we discuss in the introduction, our framework can be applied beyond the protocols we consider, such as OTR and Wire. Moreover our approach when modelling 5G handover protocols could be applied to ratcheted key-exchange or even TLS 1.3 session resumption. We leave the quantification and comparison of such – and other – protocols as future work.

Acknowledgments

Cristina Onete, Léo Robert, Pascal Lafourcade and Olivier Blazy are grateful for the support provided by the ANR through the ANR MobiS5 project (ANR-18-CE39-0019). Ioana Boureanu acknowledges the support of the EPSRC grant EP/S024565/1. This work has also been partially supported by the French government research program "Investissements d'Avenir" through the IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25), the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01), the French ANR project DECRYPT (ANR-18-CE39-0007) and SEVERITAS (ANR-20-CE39-0009).

References

[1] An open network for secure, decentralized communication, 2019. https://matrix.org/.

[2] 3GPP. System architecture for the 5G System (5GS). Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP), 10 2020. Version 16.0.0.

[3] 3GPP. Procedures for the 5g system. Technical Specification (TS) 23.502, 3rd Generation Partnership Project (3GPP), 10 2021. Version 16.7.0.

[4] J. Alwen, S. Coretti, and Y. Dodis. The double ratchet: Security notions, proofs, and modularization for the signal protocol. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 129–158, 2019.

[5] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. Internet-Draft draft-ietf-mls-protocol-14, Internet Engineering Task Force, May 2022.

[6] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 232–249, 1993.

[7] O. Blazy, A. Bossuat, X. Bultel, P.-A. Fouque, C. Onete, and E. Pagnin. SAID: Reshaping Signal into an Identity-Based Asynchronous Messaging Protocol with Authenticated Ratcheting. In *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*, pages 294–309, Stockholm, Sweden, 2019. IEEE.

[8] O. Blazy, I. Boureanu, P. Lafourcade, C. Onete, and L. Robert. How fast do you heal? A taxonomy for post-compromise security in secure-channel establishment. Cryptology ePrint Archive, Paper 2022/1090, 2022. https://eprint.iacr.org/2022/1090.

[9] N. Borisov, I. Goldberg, and E. A. Brewer. Off-the-record communication, or, why not to use PGP. In V. Atluri, P. F. Syverson, and S. D. C. di Vimercati, editors, *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, pages 77–84. ACM, 2004.

[10] C. Brzuska, E. Cornelissen, and K. Kohbrok. Security analysis of the MLS key derivation. In *43rd Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 2535–2553. IEEE, 2022.

[11] C. Brzuska, A. Delignat-Lavaud, C. Fournet, K. Kohbrok, and M. Kohlweiss. State separation for code-based game-playing proofs. In *Advances in Cryptology - ASIACRYPT Proceedings, Part III*, volume 11274 of *LNCS*, pages 222–249. Springer, 2018.

[12] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in cryptology – EUROCRYPT*, volume 2045 of *LNCS*, pages 453–474, 2001.

[13] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A Formal Security Analysis of the Signal Messaging Protocol. *Proceedings - 2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017*, (July):451–466, 2017.

[14] K. Cohn-Gordon, C. Cremers, L. Garratt, J. Millican, and K. Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In *Proceedings of ACM CCS*, page 1802–1819, 2018.

[15] K. Cohn-Gordon, C. J. F. Cremers, and L. Garratt. On post-compromise security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 164–178, 2016.

[16] B. Dowling and B. Hale. Secure messaging authentication against active man-in-the-middle attacks. In *Proceedings of EuroS&P*, pages 54–70. IEEE, 2021.

[17] J. Dreier, P. Lafourcade, and Y. Lakhnech. A formal taxonomy of privacy in voting protocols. In *2012 IEEE International Conference on Communications (ICC)*, pages 6710–6715, 2012.

[18] M. Fischlin and F. Günther. Multi-stage key exchange and the case of google's quic protocol. In *Proceedings of CCS*, page 1193–1204. ACM, 2014.

[19] R. Focardi and R. Gorrieri. A taxonomy of trace-based security properties for ccs. In *Proceedings The Computer Security Foundations Workshop VII*, pages 126–136. IEEE Computer Society, 1994.

[20] W. S. GmbH. Wire security whitepaper, 2021. https://wire-docs.wire.com/download/Wire+Security+Whitepaper.pdf.

[21] M. Marlinspike and T. Perrin. The double ratchet algorithm, 2016.

[22] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. 34, 01 2010.

A Security proofs for Signal and SAID

We consider the different cases corresponding to the behaviour of the adversary.

Conventions. We assume that all KDFs are modelled as random oracles. Each key k has $|\mathtt{k}|$ values. Note that the key space might be of same size for all keys (*e.g.*, $|\mathtt{k}|$, the order of the group for all k). The security statements are parametrized by the maximal number of stages $\mathsf{n_S}$, the maximal number of message $\mathsf{n_{x-max}}$ in a given chain, the maximal number of chain $\mathsf{n_{y-max}}$, run by any given instance, the number of parties generated by the adversary $\mathsf{n_P}$ and the number of sessions $\mathsf{n_\pi}$ created by any given party. Finally, we consider all calls to KDF as queries to random oracles.

The proofs are organized through game hops where the first game is the original security game (see Figure 3 of Sec. 2.4).

A.1 Games for Signal

$\mathbb{G}_0$: This game corresponds to the original security game (Fig. 3 of Sec. 2.4). The advantage of $\mathcal{A}$ is Adv_0.

$\mathbb{G}_1$: In this game $\mathcal{C}$ guesses P, Q, the session index of the target session, and the target stage $s^\star = (x^\star, y^\star)$ for which $\mathcal{A}$ has queried oTest.

If $\mathcal{A}$ queries another parties, session or tested stage then $\mathcal{C}$ aborts the game and returns a random bit. Therefore we have the following: $\mathsf{Adv}_0 \leq \mathsf{n_P}^2 \cdot \mathsf{n_\pi} \cdot \mathsf{n_{x-max}} \cdot \mathsf{n_{y-max}} \cdot \mathsf{Adv}_1$

The next games are dedicated to ensure that no DH values collide. Moreover, we assume the uniqueness of the identity key for each party.

$\mathbb{G}_2$: This game is the same as $\mathbb{G}_1$ except that the challenger aborts if two values ephk collide.

We have: $\mathsf{Adv}_1 \leq \binom{\mathsf{n_\pi}}{2} \cdot 2^{-|\mathsf{ephk}|} + \mathsf{Adv}_2$

At this point, the uniqueness of the master secret ms is guaranteed. Indeed, ms is computed using ik and also ephk thus by uniqueness of the former and the latter, we have uniqueness of the shared secret ms. Moreover, the sessions are also unique by uniqueness of the ephemeral keys.

$\mathbb{G}_3$: We modify the previous game to avoid collisions of honestly-generated ratchet keys rchk.

We have: $\mathsf{Adv}_2 \leq \binom{\mathsf{n_\pi} \cdot \mathsf{n_{y-max}}}{2} \cdot 2^{-|\mathsf{rchk}|} + \mathsf{Adv}_3$

$\mathbb{G}_4$: We ensure that there is no collision for honestly-generated prek. The upper-bound of total number of pre-keys is the number of sessions $\mathsf{n_\pi}$: $\mathsf{Adv}_3 \leq \binom{\mathsf{n_\pi}}{2} \cdot 2^{-|\mathsf{prek}|} + \mathsf{Adv}_4$

$\mathbb{G}_5$: The challenger needs to guess the index i of the pre-key of Q used in the tested session. Since there are $\mathsf{n_\pi}$ possible values, we have: $\mathsf{Adv}_4 \leq \mathsf{n_\pi} \cdot \mathsf{Adv}_5$

For clarity, we keep the notation prek_Q instead of prek^i_Q (signed pre-key of index i).

At this point, we will use the uniqueness and secrecy of $\mathsf{prepk}_Q^{\mathsf{rchk}_P^{0,1}}$ in order to prove indistinguishability from random of rk_1. Note that the value ms (the second input of the KDF) can be learned by any reach's adversary. [8]

$\mathbb{G}_6$: In this game the challenger accepts collision of ik_P and prek_P. We need to add this condition since the next game will use a GDH challenge where the DH pair might collide with probability $1/q$, thus: $\mathsf{Adv}_5 \leq \frac{1}{q} + \mathsf{Adv}_6$

Those previous games are shared between all possible adversaries of our model. We now partition our analysis given types of adversary.

Local Passive Outsider. In this case, the adversary can only reveal single stage keys (via the $\mathsf{oReveal.1Stage}$ oracle) in a passive way, and it has no information on the server-stored keys. Recall that Signal protocol is $(\infty, 1)$-PCS secure.

$\mathbb{G}_7$: We modify $\mathbb{G}_6$ such that the challenger aborts as soon as $\mathcal{A}$ queries the random oracle (representing the KDF) on $(\bullet \| (\mathsf{prepk}_Q)^{\mathsf{rchk}_P^{0,1}})$ where the first part of the input is analogous to ms. Since our analysis is done in the random oracle model, the only way for $\mathcal{A}$ to compute the output is to give the exact input. If so, we show that when this event occur, we can construct an adversary $\mathcal{B}$ winning a GDH challenge. Recall that the GDH experiment has input (g^a, g^b) to return g^{ab} with a DDH oracle access with input (g^x, g^y, g^z) and output 1 if $g^{xy} = g^z$.

$\mathcal{B}$ simulates $\mathbb{G}_6$ for $\mathcal{A}$ and plays against its GDH challenger. Instead of sending g^a and g^b to $\mathcal{A}$, it sends $\mathsf{rchk}_P^{0,1}$ and prek_Q

[8] ms is single stage so local, medium or global adversary can reveal it.

respectively. Notice that $\mathcal{A}$ cannot query the $\mathsf{oReveal.1Stage}$ oracle on prek_Q nor $\mathsf{rchk}_P^{0,1}$ since those keys are cross-stage keys (so $\mathcal{B}$ does not have to know the private parts of those keys). However, since $\mathcal{B}$ has replaced long-term and medium-term keys of two parties (where those keys could be used in other sessions), it must ensure a valid simulation for those (non-tested) stages. In either cases, $\mathcal{B}$ randomly chooses the value rk_1 but answers consistently with calls to the random oracle by maintaining a list. This list maps the session key with the public keys associated. Whenever $\mathcal{A}$ calls the random oracle, $\mathcal{B}$ checks if the public parts are in the list and returns the corresponding value if they are in the list, and draws a random element and adds it to the list otherwise. The special case is when $\mathcal{A}$ sends $\mathsf{CDH}(\mathsf{prepk}_Q, \mathsf{Rchpk}_P^{0,1})$ to the random oracle. In that case, the DDH oracle returns 1 when $\mathcal{B}$ queried it thus finding a solution to the GDH experiment. Finally, by noting ε_{GDH} the advantage of $\mathcal{B}$ solving the GDH problem, we have: $\mathsf{Adv}_6 \leq \mathsf{Adv}_7 + \varepsilon_{GDH}$

$\mathbb{G}_8$: This game ensures the indistinguishability of the rk/tmp outputs by the random oracle, up to and including $y^\star$. For this, we apply the modifications of $\mathbb{G}_6$ and $\mathbb{G}_7$ for a number of times equal to the maximum number of chains $\mathsf{n_{y-max}}$: $\mathsf{Adv}_7 \leq \mathsf{n_{y-max}} \cdot \left[\binom{n_\pi}{2} \cdot 2^{-|\mathsf{prek}|} + \varepsilon_{GDH}\right] + \mathsf{Adv}_8$

Note that the same argument cannot be applied to other outputs of the random oracle (such as the chain key $\mathsf{ck}^{\cdot,y^\star}$) since those values could be revealed by the adversary (which is handled in the next game).

$\mathbb{G}_9$: In this game, we ensure that the value $\mathsf{ck}^{0,y^\star}$ is unique. If there are two equal values in a session, or in two different (honest) sessions, then the challenger aborts and returns a random bit.

Recall that the random oracle model implies that a call to the KDF duplicates the output if the same inputs are used, or if true randomness repeats (with negligible probability), thus we have: $\mathsf{Adv}_8 \leq \mathsf{n_{x-max}} \cdot \binom{\mathsf{n_{x-max}} \cdot \mathsf{n_{y-max}}}{2} \cdot 2^{-|\mathsf{ck}|} + \mathsf{Adv}_9$

At this point, the chain key $\mathsf{ck}^{0,y^\star}$ is indistinguishable from random to $\mathcal{A}$ (which is due to the indistinguishability of rk/tmp values from random of $\mathbb{G}_8$).

Depending on the adversary's reach, here local, some reveal can be queried such as the chain key ck or mk. Here, the argument we used is related to the winning conditions (*i.e.*, freshness of the tested stage). Indeed, for a local passive outsider adversary, the winning conditions are parametrized by a $(\infty, 1)$ bound meaning that no $\mathsf{oReveal.1Stage}$ can be queried for a stage of index $x > 0$ and $y = y^\star - \Upsilon + 1 = y^\star$. Informally, we exclude reveal queries for stages of the same chain of the tested stage; this is a direct consequence of the symmetric ratcheting of Signal where the knowledge of one chain key implies knowledge of all the chain. Notice that our metric is also a lower bound since any strictly lower (χ, Υ) value implies $\Upsilon = 0$ meaning that $\mathcal{A}$ could reveal the chain key on a stage with $y = y^\star$. This yields a trivial attack on the session keys because of the symmetric ratcheting property.

We conclude this proof by stating that: $\mathbb{G}_9 \leq 2^{-|\mathsf{ck}|} + 2^{-|\mathsf{mk}|}$

Indeed, there are two possibilities for the adversary to recover $\mathsf{mk}^{x^\star,y^\star}$, either guessing directly this value (with negligible probability $2^{-|\mathsf{mk}|}$) or give as input to the random oracle the value $\mathsf{ck}^{x^\star-1,y^\star}$ (with negligible probability $2^{-|\mathsf{ck}|}$).

We have shown an upper bound of our metric meaning that we ensure the security for at least a given number of stages. However, the security could be *faster*, *i.e.*, find a smaller metric with unchanged security. We need to show that our metric is tight meaning that we need an extra argument to show that no security can be guaranteed with smaller metric.

In this case, a local passive outsider adversary, Signal is $(\infty, 1)$-PCS secure if we exhibit an attack which compromises at least $(\infty, 1)$ stages within the adversary's type. The attack in this case is simple, $\mathcal{A}$ can reveal $\mathsf{ck}^{1,y}$, for a given y on any peer, via the $\mathsf{oReveal.1Stage}$ oracle. This leads to compromising the full chain y because of the symmetric ratchet deriving the keys (both ck and mk). The adversary has then compromised $(\infty, 1)$ stages but no more because the next chain is initialised with cross-stage keys (*i.e.*, rchk).

Medium Passive Outsider. In this case, the adversary can reveal single and cross stages keys (via the $\mathsf{oReveal.XStage}$ oracle) in a passive way, and it has no information on the server-stored keys. Recall that (cf Fig. 4), the Signal protocol is $(\infty, 2)$-PCS secure.

$\mathbb{G}_7$: The challenger aborts if $\mathcal{A}$ gives as second input $\mathsf{CDH}(\mathsf{prepk}_Q, \mathsf{Rchpk}_P^{0,1})$ to the random oracle (the first input is a value corresponding to ms). The keys prepk_Q and $\mathsf{Rchpk}_P^{0,1}$ are now in the adversary's reach possibility, via query to $\mathsf{oReveal.XStage}$ oracle. Yet, the winning conditions of this type of adversary exclude such query for a stage of chain (minimum) index $y = y^\star - 1$ for $y > 0$. So if $\mathcal{A}$ tests a stage of index $y = 1$ or $y = 2$ then the winning conditions ruled out any call to $\mathsf{oReveal.XStage}$ for such chain. As in the local case, we show that under the GDH assumption, it holds that: $\mathsf{Adv}_6 \leq \mathsf{Adv}_7 + \varepsilon_{GDH}$

The reduction is the same as in the local case (where the reveal calls in the latter were excluded by the adversary's reach and by the winning conditions for the medium case). In this game, the advantage of $\mathcal{A}$ is the same as the local case, however the argument is different. For the local case, the adversary has no access to cross-stage keys while in this case, the adversary can query the $\mathsf{oReveal.XStage}$. Yet, the winning conditions for the medium case exclude such queries.

The rest of the proof is done the same way as for the local case, where in $\mathbb{G}_8$ the indistinguishability of rk/tmp is ensured by the winning conditions (same reason as in the previous game).

We conclude the proof by showing an attack that compromised two chains since Signal is $(\infty, 2)$-PCS secure for a medium passive outsider adversary. The adversary can reveal

$\mathsf{rchk}^{0,2}$ and rk_1 to get all the needed information to derive the keys of chain 2. It can also derive the $\mathtt{tmp}$ value used to initialise the next chain. So $\mathcal{A}$ has all the inputs to completely derive chain 3 (the other input to initialise the chain is $\mathsf{CDH}(\mathsf{rchk}^{0,3}, \mathsf{rchk}^{0,2})$ which is computable by $\mathcal{A}$).

Global Passive Outsider. This case is actually the same as for medium case. Indeed, the argument for game hops of medium adversary implies the winning conditions for indistinguishability of keys in $\mathbb{G}_6$ and $\mathbb{G}_7$. The attack exhibiting our metric can also be the same, while the global case could compromised the first two chains (while the medium adversary can only compromised chains starting from the second one). We can conclude that, for global passive outsider adversary, Signal is $(\infty, 2)$-PCS secure.

Local Active Outsider. For an active adversary, we need to ensure that the keys stored on the server are generated, and signed, by the corresponding party (and not $\mathcal{A}$). Indeed, during the registration step, a party P sends its identity key and prekeys signed with the identity key. For an active adversary, some keys might be maliciously generated and sent to the server. In the case of LAO, the adversary cannot request long-term keys from its set of oracles (only ephemeral keys). Thus, we prove that the adversary needs to forge a signature.

$\mathbb{G}_7$: Recall that in $\mathbb{G}_5$, the challenger aborts if the chosen pre-key is different from the one used in the tested session. So the reduction to the EUF-CMA game of the signature scheme is straightforward since the challenger already knows the index of the forged signature.

For the reduction, the adversary has access to a signing oracle which updates a list of keys and signatures at each call (for avoiding trivial forgery where the signature has been already queried). We denote by q_s the number of queries to this oracle. We assume here that there is an adversary $\mathcal{A}$ able to produce a valid signature on prepk (for a given index i) and we construct $\mathcal{B}$, which uses $\mathcal{A}$, to break the EUF-CMA signature scheme. $\mathcal{B}$ uses its own oracle to forward query to $\mathcal{A}$, thus: $\mathsf{Adv}_6 \leq \mathsf{Adv}_7 + \varepsilon_{EUF-CMA}$

From now, the following games are the same as in the local passive outsider.

Medium/Global Active Outsider. For those two adversaries, Signal is (∞, ∞)-PCS secure meaning that no healing is possible. So we just exhibit an attack resulting in the impossibility of PCS property. The attack is simple, the adversary injects its own initial ratchet key $\mathsf{rchk}_{\star}^{0,1}$ and reveal ms (which is in the reach's capability of medium and global adversaries) during the initialisation phase. Thus $\mathcal{A}$ hijack the communication, where Bob is convinced to communicate with Alice, but Alice has no access to the communication (since the chain keys are different). In this case, Bob will continue the communication as long as $\mathcal{A}$ is following the protocol (it does not need to deviate from the protocol anymore).

Passive Insider. Each of those three types of adversaries (local, medium and global) corresponds to passive outsider adversaries. Indeed, the difference between outsider and insider is that the latter poses as the super user $\hat{S}$. In the case of Signal, this corresponds to the semi-trusted server which receives the public *bundle* keys upon registration. Because of the passive access type, the adversary cannot interfere with those keys so there is no difference with outsider adversary (the public keys stored by the server are also accessible by outsider adversary). For Signal, there is no difference between passive insider and passive outsider given the reach capability (local, medium, global). For this reason, the metric is the same for local, medium or global between outsider and insider, in the passive access type.

Active Insider. The case of active insider is the strongest type of adversary. Indeed, $\mathcal{A}$ can interfere with the protocol (*e.g.*, stop, modify messages) while compromising the server. This critical case (either local, medium or global) cannot include healing as the adversary can manipulate the keys from the start of the communication. An active insider adversary can simply remove honestly-generated keys sent to the server and replace them by its own malicious key material. In this case, the adversary plays a PiTM (Personn in The Middle) forwarding messages through Alice to Bob (and vice-versa) by its own. The communication between Alice and Bob cannot heal thus leading to a (∞, ∞)-PCS security for active insider adversary.

A.2 Games for SAID

$\mathbb{G}_0$: This game corresponds to the original security game (Fig. 3 of Sec. 2.4). The advantage of $\mathcal{A}$ is Adv_0.

$\mathbb{G}_1$: In this game $\mathcal{C}$ guesses P, Q, the session index of the target session, and the target stage $s^\star = (x^\star, y^\star)$ for which $\mathcal{A}$ has queried oTest.

If $\mathcal{A}$ queries another parties, session or tested stage then $\mathcal{C}$ aborts the game and returns a random bit. Therefore we have the following: $\mathsf{Adv}_0 \leq \mathsf{n_P}^2 \cdot \mathsf{n}_\pi \cdot \mathsf{n_{x-max}} \cdot \mathsf{n_{y-max}} \cdot \mathsf{Adv}_1$

We assume the uniqueness of the identity key, and identity-based related keys (identification and signature ones) for each party. The latter condition is ensured by the KDC maintaining a list of keys and removing possible duplicates.

Local Passive Outsider. We prove that SAID has the best healing, *i.e.*, $(1, 0)$-PCS security meaning that only the compromised stage is accessible to the adversary. Recall that a local adversary can query the $\mathsf{oReveal.1Stage}$ oracle to reveal single-stage keys, which for SAID correspond to ck and mk. First we show that the master secret ms is indistinguishable from random, then we show that a tested stage is fresh (with an indistinguishable session key from a random value) even after an immediate compromised stage.

$\mathbb{G}_2$: This game aborts if the adversary calls the random oracle with input ms_{PQ}. The adversary has only one value to guess, value r while other values are already determined (the identity of Q, and the master public key of $\hat{S}$). Guessing r corresponds to a large failure event so: $\mathsf{Adv}_1 \leq \frac{1}{q} + \mathsf{Adv}_2$

From this game, we assume that the master secret is unique between each pair of communicating partners.

$\mathbb{G}_3$: We show that $\mathsf{ck}^{1,y}$ is indistinguishable from random. In this game, the challenger aborts if the adversary query the random oracle with input $(\bullet\|\Delta^\star\|\mathsf{rk}^\star)$ where $\bullet$ corresponds to ms. We show by reduction to GDH that if $\mathcal{A}$ can query the random oracle with $\Delta^\star = DH(\mathsf{Rchpk}^{0,y^\star}, \mathsf{Rchpk}^{0,y^\star+1})$ with non-negligible probability then we can construct $\mathcal{B}$ breaking the GDH problem. We apply the same technique as in Signal (cf. $\mathbb{G}_7$), that is $\mathcal{B}$ sends $\mathsf{Rchpk}^{0,y^\star} := g^a$ and $\mathsf{Rchpk}^{0,y^\star+1} := g^b$ to $\mathcal{A}$. If $\mathcal{A}$ does not send the query corresponding to $\Delta = \Delta^\star$ then $\mathcal{B}$ simulates completely the game for $\mathcal{A}$ while the special case is when $\mathcal{A}$ sends $\mathsf{CDH}(\mathsf{Rchpk}^{0,y^\star}, \mathsf{Rchpk}^{0,y^\star+1})$ to the random oracle. In this case, when $\mathcal{B}$ queries its DDH oracle (returning 1) it finds a solution to the GDH experiment. Finally, we have: $\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \varepsilon_{GDH}$

$\mathbb{G}_4$: This game is the same as the previous except that the challenger aborts if $\mathcal{A}$ queries the random oracle with rk for up to and including $y^\star$. We use hybrid argument where the first game is $\mathbb{G}_4$ and each iteration are the next rk until $\mathsf{rk}^\star$. Between each game, the root keys are indistinguishable since the new root key is the output of the random oracle and $\mathcal{A}$ can only query $\mathsf{oReveal.1Stage}$. Since the adversary's probability to guess the root key is $2^{-|\mathsf{rk}|}$ for $\mathsf{n_{y-max}}$ number of chains, we have: $\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \frac{1}{2^{-|\mathsf{rk}|}-1}$

$\mathbb{G}_5$: This game aborts if the adversary queries the random oracle with $(\mathsf{ck}^\star, \mathsf{ms}_{PQ}, \mathsf{tag}^\star)$. This game proceeds as the previous one with a subcase to handle. Indeed, key ck is in the adversary's reach (single-stage for a local adversary). Thus $\mathcal{A}$ could reveal this keys by querying oracle $\mathsf{oReveal.1Stage}$. However, as defined in 2.4, the adversary wins with non-negligible probability for a query on stage $s^\star$ which is the tested stage. Yet, the adversary has negligible probability to win if the tested stage is after the reveal query. Indeed, SAID is $(1,0)$-PCS secure meaning that $\mathcal{A}$ could reveal a key on stage $x^\star - 1$ but distinguishes the session key with negligible probability.

Suppose that $\mathcal{A}$ does not query $\mathsf{oReveal.1Stage}$ on the tested stage (which is part of our metric definition). We show by reduction that $\mathcal{A}$ can distinguish the session key if it can break the BCDH assumption meaning that it can compute ms_{PQ}. We construct $\mathcal{B}$ simulating the game for $\mathcal{A}$. Adversary $\mathcal{B}$ receives $A = g_1^a, B = g_2^b, C = g_2^c$ as input. It sets $H(R) := A$ (with H simulated as random oracle and R the responder role the tested session) and $\mathsf{ID.mpk} := B$. For each other party $X \neq R$, $\mathcal{B}$ generates a random value α and sets $H(X) := g_1^\alpha$. When $\mathcal{A}$ starts the session between P and Q then $\mathcal{B}$ runs the actual protocol except that it sets $h := C$. In this case, we have $\mathsf{ms}_{PQ} = e(A,B)^c$ which is the solution of the BCDH problem instance. Observe that $\mathcal{B}$ simulates perfectly the game, except when $\mathcal{A}$ sends ms_{PQ} to the random oracle. Thus we have: $\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \varepsilon_{BCDH}$

Finally, if $\mathcal{A}$ never sends ms_{PQ} to the random oracle then the session key is indistinguishable from random. In this case, $\mathcal{A}$ wins the game with probability $1/2$: $\mathsf{Adv}_5 = \frac{1}{2}$

Local Active Outsider. This case is the same as the passive adversary except that we need to ensure that the adversary cannot replace ms with its own key material. Here, the adversary has two possible ways to inject its own key material. First, $\mathcal{A}$ could interfere during the registration phase between P and $\hat{S}$. However, we assume that those two parties establish a secure channel thus the security relies on the AKE assumption. Second, $\mathcal{A}$ could forge its own value h to compute the master secret but in this case, we rely on the EUF-CMA security of the IB-signature scheme IBSig. Note that the active adversary case cannot interfere later on because the other keys are not single-stage (thus having the same security as the local case).

Medium/Global Passive Outsider. This case gathers both medium and global adversaries. Indeed, a global adversary has additional access to the keys used during registration (ik) and identity-based key used for instance initialisation. However, in the passive this yields to no other consequence than the medium case.

SAID is $(\infty,2)$-PCS secure meaning that the adversary can compromised two full chains of communication. We apply the same game hops as the local case, but the index of stages are different. Indeed, our security definition ensures that the call to $\mathsf{oReveal}$ cannot happen with stage $y = y^\star - 1$ or $y = y^\star$. Thus from $y = y^\star$, the adversary has the same advantage of the local case.

Local Passive Insider. In this case, $\mathcal{A}$ can reveal ms_{PQ} but not inject its own value during the protocol. We show that such adversary can compromise at most the first chain which corresponds to $(\infty,1)$-PCS security. This is due to the fact that a new chain is initialised with ratchet keys which are out of reach's adversary.

$\mathbb{G}_2$ This game aborts if the adversary queries the random oracle with input $(\bullet\|\Delta^\star\|\circ)$ where $\bullet$ corresponds to ms and $\circ$ corresponds to $rk^\star$. We apply the same argument as the local passive outsider adversary in $\mathbb{G}_3$. Thus we also use GDH reduction to show that $\mathcal{A}$. The rest of the proof correpsonds to the local passive outsider since at this point the adversary has the same advantage in both cases.

Medium/Global Passive Insider. In this case, SAID is $(\infty,2)$-PCS secure. This comes directly from the fact that now the adversary has access to the secret keys used during session initialisation but cannot interfere in other way with the protocol. Our security definition implies that the adversary cannot compromise a stage of index $y = y^\star - 1$ or $y = y^\star$. We apply then the same proof as the previous case (local passive insider).