# Falsifiability, Composability, and Comparability of Game-based Security Models for Key Exchange Protocols

**Version 12.0, July 2024**[⋆]

Chris Brzuska[1], Cas Cremers[2], Håkon Jacobsen[3],
Douglas Stebila[4], and Bogdan Warinschi[5]

[1] Aalto University
[2] CISPA Helmholtz Center for Information Security
[3] University of Oslo
[4] University of Waterloo
[5] University of Bristol & DFINITY

**Abstract.** A security proof for a key exchange protocol requires writing down a security definition. Authors typically have a clear idea of the level of security they aim to achieve. Defining the model formally additionally requires making choices on games vs. simulation-based models, partnering, on having one or more Test queries and on adopting a style of avoiding trivial attacks: exclusion, penalizing or filtering. We elucidate the consequences, advantages and disadvantages of the different possible model choices.

Concretely, we show that a model with multiple Test queries composes tightly with symmetric-key protocols while models with a single Test query require a hybrid argument that loses a factor in the number of sessions. To illustrate the usefulness of models with multiple Test queries, we prove the Naxos protocol security in said model and obtain a tighter bound than adding a hybrid argument on top of a proof in a single Test query model.

Our composition *model* exposes partnering information to the adversary, circumventing a previous result by Brzuska, Fischlin, Warinschi, and Williams (CCS 2011) showing that the *protocol* needs to provide public partnering. Moreover, our baseline theorem of key exchange partnering shows that partnering by *key equality* provides a joint baseline for most known partnering mechanisms, countering previous criticism by Li and Schäge (CCS 2017) that security in models with existential quantification over session identifiers is non-falsifiable.

## 1 Introduction

Key exchange protocols are at the heart of most secure real world communication protocols: they establish a shared secret session key for further use, typically a symmetric-key based secure channel. Additionally they usually also provide a form of authentication, referred to as *authenticated key exchange* (AKE).

Given the ubiquity of AKE protocols, one might expect that they meet a well-understood and standard security notion. Unfortunately, this is not the case: security models for AKE protocols are surprisingly diverse and complex, mainly caused by a large range of choices for functionalities and adversary capabilities.

Examples of functionality choices are the number of parties that can jointly establish a key; whether authentication is based on public keys, shared symmetric keys, or passwords; and whether parties know their intended peer's identity in advance or if it is learned during the execution of the protocol. Combined with a given security property, such as unilateral or mutual authentication, this provides the basic constraints for an AKE protocol.

Ideally, one then shows that a protocol additionally offers strong security guarantees. This involves modeling the adversary's capabilities, such as the secret information that the adversary can learn, and at what time; whether it can reveal internal state, subvert random number generators, register malicious keys, obtain long-term keys through legal means, and so on. There are inherent trade-offs between achieving protection against these, and it impacts efficiency, so different contexts require different answers.

In the literature, such aspects have been incorporated in different ways into monolithic security models, leading to a very wide range of different models, see [17, Ch. 2] for an extensive introduction. Defining security requires authors to make a multitude of additional technical choices, many of which are valid and can be justified. As a result, there need not necessarily be a single best key exchange model.

[⋆] We provide a changelog in Appendix C.

**Contributions.** It is not our goal to classify models as either good or bad. Instead, we aim to provide conceptual discussions and formally substantiated insights into the consequences of different modeling choices, notably the following four:

1. Game-based vs. simulation-based security: impact on composability.
2. Partnering definitions (i.e., how a model defines what the partner session is that is expected to compute the same key): impact on composability, falsifiability, comparability and agreement properties.
3. Single vs. multiple test queries: impact on the tightness of composition proofs.
4. Exclusion vs. penalizing vs. filtering of trivial attacks: impact on expressibility, comparability, strength and (subjective) clarity of the model.

We expand on these aspects in Sections 1.1 to 1.4, and illustrate how existing models for key exchange instantiate them in Table 1.

Having explored various consequences of these modeling choices, we then proceed to:

– develop a security model for key exchange protocols that deals consistently with each of the above four choices;
– prove a general composition theorem that shows how to compose a key exchange protocol with symmetric-key protocols, with weaker preconditions than previous results; and
– prove a lemma on key exchange partnering that partnering by *key equality* is a joint baseline for most models.

As a case study for some of our model choices, we prove the Naxos protocol secure with a tighter bound than in the original paper.

### 1.1 Game-based vs. Simulation-Based security

While it was originally thought that simulation-based security is inherently more composable than game-based security, the lines between the two notions have become blurred over the years. In the end, a security game can often be seen as an *instantiation* of a simulation-based definition. For example, for key exchange security, the simulation-based definition postulates the existence of a simulator which simulates a protocol run independently of the key, whereas the game-based security notion can be seen as using a standard protocol run (independently from the session key) as a simulation. Indeed, Canetti and Krawczyk [26] show that Bellare–Rogaway secure key exchange protocols imply (a variant of) UC-security and thus compose securely with other UC-protocols. Brzuska, Fischlin, Warinschi and Williams [22] later showed that Bellare–Rogaway secure key exchange protocols compose securely also with arbitrary symmetric-key-based games. Nowadays, there is a plethora of variants of game-based key exchange definitions with varying levels of strength, each of which would require its own composition theorem. In this paper, we show via a unified composition theorem that our security model composes securely with symmetric-key protocols; as our security model is parameterizable to capture different variants, this yields composability for all variants in our model.

### 1.2 Partnering Mechanisms

Partnering mechanisms are used in AKE security models to identify *related* sessions. Most AKE security models give the adversary the power to reveal session keys, but only of sessions *unrelated* to the target session that the adversary is supposed to attack (by trying to distinguishing its real session key from random). If we want any protocol to be secure with respect to a model, the adversary cannot be allowed to reveal the session key of the target session itself, since this is the real key by design. But there is another session that also ought to be off-limits: the one residing at the target session's communication peer in a normal protocol run, because by correctness, this session *should* compute the same key as the target session. To model this, we must be able to identify this partner session within the model. This is one of the main purposes of the partnering mechanism.[6]

We can divide security models into two main classes based on their partnering mechanism: ones using *universal* partnering, and ones using *existential* partnering. For models based on universal partnering there is a single fixed partnering mechanism used for all protocols within the model, and a protocol is considered secure in the model if it can be proven secure with respect to this single mechanism. On the other hand, for models based on existential partnering, there is not a single fixed partnering mechanism. Instead, a protocol is considered secure in the model only if there *exists* a partnering mechanism for which the protocol can be proven secure.

[6] The partnering mechanism is also sometimes used to define authentication, e.g., authentication is broken if there exists a session that accepted without a partner (provided the intended peer's authentication credential was uncompromised), or used to determine if the attacker was temporarily passive. For now, we focus on partnering for defining session key indistinguishability, and deal with authentication separately later.

*Universal partnering.* The most common form of universal partnering is *matching conversations*, where two sessions are partners if they have the same transcript of messages sent and received (except possibly the last message which may have been dropped by the adversary); this partnering mechanism originates from the first AKE security model by Bellare and Rogaway [10][7]; a small sample of important models that employ it include [13, 15, 30, 32, 44, 48, 53, 56]. Other forms of universal partnering include key-partnering [42, 47] and original-key-partnering [43, 55].

*Existential partnering.* In their second AKE paper, Bellare and Rogaway [11] introduced existential partnering via *partner functions*. In this model a protocol is secure if there exists some partnering function for which it can be proven secure. It is up to the prover to demonstrate such a partner function exists, either by construction or by proving that it must exist. Other papers working with partnering functions generically include [19, 61]. Models using *session identifiers* (SIDs) [3, 9, 21–25, 34, 38, 46, 50, 59] can also be seen as using existential partnering. Specifically, the SID can be viewed as a function that simply computes the SID string based on various inputs (including, e.g., a partial transcript, secret information, or externally provided information). Of course, matching conversations, and any of the other universal partnering mechanisms, are also suitable, and thus security models using partner functions can be seen as a generalization of security models using universal partnering.

*Falsifiability.* We refer to the process of showing that a protocol is insecure in a model as *falsifiability*: one falsifies the security of a protocol (with respect to a model, and the model should support this possibility). To show that a protocol is insecure in a model with universal partnering, one can demonstrate an attack that breaks security with respect to the partnering mechanism specified by the model.

On the other hand, to show that a protocol is insecure in a model with existential partnering, one would have to demonstrate attacks that break security for all possible partnering functions, which could be substantially harder. How, then, should one interpret the absence of a proof of security with respect to a particular session identifier or partner function? Is it possible to demonstrate an attack against the same protocol in the same model with respect to a different session identifier? What constitutes a *good* session identifier? And how do we know whether a protocol is insecure or whether we merely failed to find a suitable session identifier?

Thus, models with universal partnering offer a more direct path to falsifiability. However, partnering using, e.g., matching conversations could be too strong a requirement, since adding an irrelevant bit to a secure protocol suddenly makes it insecure, defying our intuition. Indeed, Li and Schäge [55] showed a class of "no-match" attacks that demonstrated flaws in several existing security proofs in models based on matching conversations.

One alternative universal partnering mechanism is *key partnering* [42, 47], where two sessions are considered partners if they have computed the same session key. Li and Schäge [55] however argue that *original-key partnering* should be used, where two sessions are considered partners if they would have computed the same session key in the presence of a passive adversary.

*Key partnering.* We show that key partnering is a universal choice for AKE partnering mechanisms, together with a partnering oracle that is added to the model and allows the adversary to determine whether two sessions have the same key. From the perspective of writing proofs, key partnering is relatively easy to work with, especially compared to original-key partnering.

Importantly, we demonstrate in Section 3 that key partnering comes with strong falsifiability properties: we show that if an attack is shown against a protocol in a model where key partnering is used, then that attack would be present in the model with respect to *any* valid partnering mechanism. We prove this falsifiability result by establishing the contrapositive. Namely, we show what we call the *baseline theorem of partnering in key exchange*: for any fixed freshness condition, if a key exchange protocol provides key indistinguishability in the model using a partnering mechanism that satisfies certain soundness properties, then that protocol is also secure in the model using key partnering.

A consequence of our baseline theorem of key exchange partnering is that researchers retain the option of proving security using session identifiers or other partner functions, but can be assured of falsifiability by also proving that their partner function satisfies the soundness conditions required by the baseline theorem.

*Composability.* Our aforementioned composability result for key agreement protocols with symmetric-key protocols indeed relies on *key partnering*, sidestepping a seeming no-go result of Brzuska, Fischlin, Warinschi, and Williams [22], which states that composability requires the *protocol* to provide public partnering mechanism. We circumvent their impossibility result by giving the adversary an oracle to learn which two sessions are partnered. This only makes the model stronger and simplifies composition, as we will show in Section 4.2.

[7] Note that the Bellare and Rogaway version also required strict temporal ordering of the individual messages sent and received, excluding e.g. pre-plays; this requirement was dropped by later works.

*Comparability.* The central benchmark around which to compare models is normally the capabilities granted to the attacker. For example, a model in which the adversary can learn both the parties' long-term keys *and* the sessions' internal randomness ought to be stronger than a model where it can only do the former. Of course, not all models are formally comparable. For instance, a model where only long-term keys can be revealed is not necessarily stronger nor weaker than a model where only internal randomness can be revealed. But even in the case where two models do provide the same attacker queries, they can still differ in strength due to how the access to these queries are controlled, i.e., they have different freshness conditions.

Cremers and Feltz [32,37] formalized these observations, defining a security model to be precisely the collection of adversary capabilities (queries) and a freshness condition. They could then compare the relative strengths of a large number of models. However, it turns out that there are further factors that can additionally influence the comparability between models, and which were not covered by Cremers and Feltz [32]. The first is the choice of partnering mechanism. Indeed, Cremers [31] shows that the extended Canetti–Krawczyk (eCK) model [52] is in fact formally incomparable with the original CK model [24] – partially due to a mismatch in the partnering definitions.

*Agreement properties.* The model we develop, besides key indistinguishability, also covers agreement and authentication properties. These properties provide guarantees of the following form: if two parties agree on a session key, then they also agree on various other variables determined during the protocol run. Examples of such variables include: party identities, communication roles, and negotiated ciphersuites. An easy way to achieve such a property in practice is by hashing the entire transcript into the key. In the case such a practice is adopted, the notions key partnering and matching conversations coincide under the assumption that the key derivation function is collision-resistant. Transcript hashing has been adopted in TLS 1.3 and is generally considered good practice. In addition, the parties might hash further agreement data into their key. In this case, agreement on the transcript alone does not imply agreement on further variables, while agreement on the key still implies their equality.

### 1.3 Single vs. Multiple Test Queries

From the perspective of *building* an AKE protocol, we want a security model that facilitates a tight reduction from the security of the AKE protocol to the security of its underlying primitives. Later works have started to develop tight reductions of this form [4, 5, 29, 43].

In addition, from the perspective of *using* an AKE protocol, we also want a security model that facilitates a tight reduction from the composition "AKE protocol + symmetric-key protocol" to the security of the underlying AKE protocol. It is this composition we focus on here. Unfortunately, traditional models tend to lose at least a factor in the number sessions, sometimes even a square. To see why, recall that key exchange models like the BR model [10] typically have a single Test query. This means that in the composition proof we cannot replace the session keys all at once, but instead need to employ a hybrid argument where they are replaced one-by-one (as illustrated by the proof of the composition result in [22]). Indeed, this incurs a tightness loss [27, 28] in the number of session keys replaced. If the AKE protocol itself had a non-tight proof with a linear loss in the number of sessions $n$, then we are now up to an $O(n^2)$ loss of tightness for the whole composition; in real-world protocols like TLS, the number of sessions could be on the order of billions, so an $n^2$ tightness loss would have a substantial impact on the selection of parameters.

It seems more useful if an AKE model supports tight composition with the symmetric-key protocol, and the natural response is to allow *multiple* Test queries. But how should these Test queries be answered? Should each query be answered independently as real-or-random (and the adversary wins if it can distinguish at least one query), or should all answers be either all-real or all-random?

Let's call the first approach $n$-FtG (for Find-then-Guess) and the second RoR (for Real-or-Random), where $n$ is the number of Test queries. Unfortunately, $n$-FtG is no better than 1-FtG for the purposes of composability: even if $n$-FtG has multiple Test queries, during the composition proof they cannot be used to replace the session keys with random all at once, since each query is independently answered with either real or random.

In contrast, the RoR notion *does* allow all keys to be replaced all at once during the composition proof, resulting in no additional security loss for the combined construction (we will make this more precise in Section 5). The conclusion is that RoR is the most appropriate model to use for composing AKE protocols, justifying its use in our model in Section 2.

Note that switching from 1-FtG to RoR does not necessarily move the tightness problem elsewhere in the chain of results: perhaps surprisingly, our case study of NAXOS+ in Appendix A achieves the same advantage bounds in RoR as the original proof in the 1-FtG notion. In general, we suggest the use of multiple Test queries with the same secret bit to enable tightness of reductions.

### 1.4 Misbehaving adversaries: exclusion, penalizing, and filtering

A key exchange model needs to define which adversaries are considered valid. Specifically, the freshness condition defines the class of misbehaving adversaries as those that trivially win by violating the predicate. Naturally, we only want to measure the success probability of non-misbehaving adversaries. In the literature, there are essentially three ways to do this: (a) the *exclusion* approach [8], in which one only quantifies over the class of valid adversaries; (b) the *penalizing* approach [19], where, posteriori, misbehaving adversaries are penalized for their actions; and (c) the *filtering* approach [42, 60] where responses to misbehaving adversaries are silenced.

*Model strength and comparability.* We argue that security with respect to the *filtering* approach implies security with respect to the *exclusion* approach, while the implication in the other direction is false. The reason is that an adversary which is valid according to the exclusion approach is valid with respect to the filtering approach, but not vice versa. Additionally, the adversary might learn additional information via the filtering feedback which it receives from the model. By the principle of choosing the stronger model when in doubt, it seems useful to deploy the filtering approach. In addition, this means that one's security statement is at least as strong as those made by others, all other things being equal (which, admittedly, is rarely the case in key exchange models).

*Expressibility.* It turns out that the penalizing approach is somewhat incomparable to filtering. Namely, eCK-security inherently relies on the penalizing approach: The adversary is first permitted to Test as session, even though the game does not know yet whether this session is fresh or not. The adversary is then penalized in case it turns out the session does not become fresh. An analogous mechanism cannot be achieved via exclusion or filtering. If such after-the-fact-freshness properties are needed, one has to adapt a penalizing or exclusion approach. In all other cases, filtering seems to be the preferred option which is why we adopt it in our model family.

## 2 Security model

In this section we specify a parameterized model that defines a family of key exchange models. Our models can capture a variety of relevant security properties within the same carefully constructed formalism that results in security definitions that are comparable (at least amongst each other), support falsifiability and tight composition results. To formalize this family, we employ a two-step approach. First, we abstract the main security goal into a *security predicate*, and give predicates representing common key exchange security goals, such as session-key indistinguishability (in Appendix B we also address authentication security goals). Second, we allow a *freshness condition* to refine the security goal to capture security against different attacker models such as forward secrecy, which also simplifies the comparison of models, in the spirit of works such as [16, 36].

Both the issue of partnering and how misbehaving adversaries (cf. Section 1.4) are handled, are in some sense technicalities. Unlike the attacker capabilities and freshness predicates, they do not correspond to our intuitive idea of model strength. Thus, an essential step in facilitating comparison between models is fixing as many of the components of the model as possible, with the only variable being the freshness condition encoding different attacker capabilities. But what to fix these to? In particular, which partnering mechanism should you choose, and how should you capture misbehaving adversaries?

Following our earlier observations, we fix the partnering mechanism to key partnering. Our baseline theorem of key exchange partnering implies that this choice does not sacrifice comparability, as long as the partnering mechanism satisfies the required conditions.

As for dealing with a misbehaving adversary, we suggest the filtering approach which makes rules of accepted behavior explicit in the game code (see Figure 2, lines 307–311) and yields monotonic winning conditions. We showcase the filtering approach in our case study (Appendix A), where we encode game hops by successively modifying the game's filter function with each hop, until the adversary can, information-theoretically, not win anymore.

For maximal comparability, we recommend encoding the filtering rules as publicly checkable predicates, which makes exclusion-style and filtering-style definitions equivalent (see Section 6). The IsPartnered oracle we use in our general security experiment is an instance of such a public encoding: it allows publicly checking whether the adversary may reveal a session key or not.

We begin with the abstract algorithms ("syntax") that we use as the interface to a key exchange model and correctness thereof. Next we describe the AKE security experiment, which is parameterized over a freshness condition and a security predicate. We can then define the security properties of key indistinguishability and key confinement. Properties related to authentication are given in Appendix B.

*Notation.* $y \leftarrow A(x)$ denotes running a deterministic algorithm $A$ with input $x$, and storing the output in the variable $y$. Similarly, $y \leftarrow\!\!\$\ A(x)$ denotes running a probabilistic algorithm $A$ with (implicit) uniform random coins. We often use superscript to indicate function parameters, e.g., $A^{O}(x)$ to denote an algorithm with access

**Table 1:** Various AKE security models in terms of our four characteristics

| Model | Partnering Mechanism | Existential / Universal | Public Partnering | Test's Hidden Bit | Parameterizable Freshness | Adversary Behaviour |
|---|---|---|---|---|---|---|
| *BR family* | | | | | | |
| BR93/BWM [10, 15] | matching conversations | U | ● | 1-FtG | ○ | penalize |
| BR95 [11] | partner function | E | ● | 1-FtG | ○ | exclude |
| BPR [9] | session identifiers + key partnering | E | ○ | 1-FtG | ○ | penalize |
| AFP [3] | session identifiers | E | ◓ | RoR | ○ | filter |
| KSS [47] | key partnering | U | ○ | 1-FtG | ○ | penalize |
| Tight (BHJKL) [4] | matching conversations | U | ● | $n$-FtG | ○ | penalize |
| Tight (CCGJJ) [29] | matching conversations | U | ● | RoR | ○ | exclude |
| *CK family* | | | | | | |
| CK01 [24] | session identifiers | E | ◓ | 1-FtG | ○ | exclude |
| $\text{CK}_{\text{HMQV}}$ [48] | transcript | U | ● | 1-FtG | ○ | exclude |
| *eCK family* | | | | | | |
| eCK [52] | transcript | U | ● | 1-FtG | ○ | penalize |
| MU08 [56] | matching conversations | U | ● | 1-FtG | ○ | penalize |
| eCK-PFS, $\text{eCK}^w$ [32] | transcript + origin sessions | U | ● | 1-FtG | ● | exclude |
| *"Darmstadt family"* | | | | | | |
| Composable BR [22] | public session identifiers | E | ● | 1-FtG | ○ | penalize |
| Less is more [21] | session identifiers | E | ◓ | 1-FtG | ○ | penalize |
| State-separating proofs [18] | partner functions | E | via oracle | RoR | ○ | exclude |
| TLS 1.3 [34] | session + contributive identifiers | E | ○ | $n$-FtG | ○ | penalize |
| *Others* | | | | | | |
| George–Rackoff [42] | key partnering | U | via oracle | 1-FtG | ○ | filter |
| ASICS [16] | partner function | E | ◓ | 1-FtG | ● | exclude |
| Li–Schage [55] | original-key partnering | U | ◓ | 1-FtG | ○ | — |
| **This paper** | **key partnering** | **U** | **via oracle** | **RoR** | ● | **filter** |

Legend: ● yes; ◓ not necessarily; ○ no; — not applicable

to oracle $O$. We write $A(x) \mapsto y$ when presenting the type of $A$: $A$ takes arguments $x$ and yields $y$, after which we describe $A$'s domain (for $x$) and range (for $y$). We denote by $L = [x_1, \ldots, x_n]$ that $L$ is a list of $n$ elements, where $L[i]$ denotes its $i$-th element. We write $L \twoheadleftarrow x$ to denote appending the element $x$ to the list $L$, or adding $x$ to the set $L$. We write $L_1 \| L_2$ to denote the concatenation of two lists. We also write $L[x]$ to denote the entry for key $x$ in the dictionary $L$. Party identities are elements of $\mathbb{N}$. The equality test $\equiv$ treats two values as equal only if they have previously been defined, i.e., $x \equiv y \Leftrightarrow (x = y) \wedge (x \text{ and } y \text{ are defined})$.

### 2.1 Syntax of key exchange

As noted above, for simplicity we focus on two-party key exchange algorithms authenticated using public keys.

**Definition 1 (Key exchange protocol).** *A* key exchange protocol *is a tuple of algorithms* $\Pi = (\text{KG}, \text{New}, \text{Run})$*:*

- $\text{KG}() \mapsto (\mathsf{sk}, \mathsf{pk})$*: a probabilistic* long-term key generation algorithm *that outputs a private/public key pair* $(\mathsf{sk}, \mathsf{pk})$.
- $\text{New}(U, \mathsf{sk}_U, \mathsf{pk}_U, \mathsf{role}, V, \mathsf{PK}) \mapsto (\pi, m)$*: the probabilistic* protocol activation algorithm *takes as input the long-term key pair* $(\mathsf{sk}_U, \mathsf{pk}_U)$ *of party* $U$*, its role* ($\mathsf{role}$) *in this protocol run, its intended peer* $V$ *(or an empty value* $\star$*), and a dictionary* $\mathsf{PK}$ *of all parties' long-term public keys, indexed by party identity. It outputs a new instance state* $\pi$ *(defined next) at party* $U$ *and a (possibly empty) outgoing initial message* $m$*.*
- $\text{Run}(\pi, m) \mapsto (\pi', m')$*: the deterministic* protocol execution algorithm *takes as input an instance state* $\pi$ *and an incoming message* $m$*, and outputs an updated state* $\pi'$ *and (possibly empty) outgoing message* $m'$*.*[8]

We allow each party $U$ to run multiple instances ("sessions") of the protocol; all data related to a specific instance is recorded in an *instance state* $\pi$, which contains the following variables set by New:

- $\pi.\mathsf{owner}$: the party to which the instance $\pi$ belongs
- $\pi.\mathsf{sk}, \pi.\mathsf{pk}$: the long-term private/public key pair of party $\pi.\mathsf{owner}$
- $\pi.\mathsf{role}$: the role of this party in this run of the protocol, either $\mathsf{init}$ or $\mathsf{resp}$
- $\pi.\mathsf{peerID}$: the party identity of $\pi$'s intended peer

[8] Run is deterministic; all per-instance randomization is incorporated in the instance variable $\pi.\mathsf{rand}$ generated during the New algorithm.

– $\pi$.PK: the dictionary of public keys
– $\pi$.status: $\pi$'s status: running, accepted, or rejected
– $\pi$.transcript: list of all messages sent and received by $\pi$ in chronological[9] order
– $\pi$.rand: randomness used by $\pi$
– $\pi$.k: the session key derived by $\pi$. If no key has been derived yet, we use the symbol $\perp$; when we compare the session keys of two sessions, if both are $\perp$, we will not consider those session keys to be equal

A key exchange protocol is *correct* if, when messages are relayed faithfully between two honest sessions, both sessions accept, compute the same session key, and have each recorded the other as its peer. Note that we have two versions of the correctness definition, one for protocols that allow post-specified peers (i.e., the responder learns its intended peer's identity during the execution of the protocol), and one for protocols only allowing pre-specified peers (the responder must be initialized with its intended peer's identity at the beginning of the session).

**Definition 2 (AKE correctness, post-specified peer model).** *A $k$-message AKE protocol $\Pi$ allowing post-specified peers is $\varepsilon$-correct if for all $U, V \in \mathbb{N}$, and all $u \in \{U, \star\}$, we have*

$$\Pr[\mathsf{Corr}_{\Pi,U,V,u}() \Rightarrow 1] \geq 1 - \varepsilon \ ,$$

*where* Corr *is the experiment defined in Figure 1.*

AKE correctness for the pre-specified peer model is as Definition 2 except for requiring that the responder is initialized with $u = U$.

$\mathsf{Corr}_{\Pi,U,V,u}()$
101 // Set up long-term key pairs
102 $(\mathsf{sk}_U, \mathsf{pk}_U) \leftarrow\!\!\$\ \Pi.\mathsf{KG}(), \mathsf{PK}[U] \leftarrow \mathsf{pk}_U$
103 $(\mathsf{sk}_V, \mathsf{pk}_V) \leftarrow\!\!\$\ \Pi.\mathsf{KG}(), \mathsf{PK}[V] \leftarrow \mathsf{pk}_V$
104 // Initialize initiator and responder sessions
105 $(\pi, m) \leftarrow\!\!\$\ \Pi.\textsc{New}(U, \mathsf{sk}_U, \mathsf{pk}_U, \mathsf{init}, V, \mathsf{PK})$
106 $(\pi', \perp) \leftarrow\!\!\$\ \Pi.\textsc{New}(V, \mathsf{sk}_V, \mathsf{pk}_V, \mathsf{resp}, u, \mathsf{PK})$
107 // Relay messages back and forth between initiator and responder,
108 // updating their respective states $\pi$ and $\pi'$
109 **for** $i \leftarrow 1$ **to** $\lfloor k/2 \rfloor$**:**
110 $\quad (\pi', m') \leftarrow \Pi.\textsc{Run}(\pi', m)$
111 $\quad (\pi, m) \leftarrow \Pi.\textsc{Run}(\pi, m')$
112 **if** $k$ odd**:**
113 $\quad (\pi', \perp) \leftarrow \Pi.\textsc{Run}(\pi', m)$ // initiator sends the last message
114 // Check correctness condition
115 **return** $(\pi.\mathsf{status} = \mathsf{accepted}) \wedge (\pi'.\mathsf{status} = \mathsf{accepted}) \wedge$
116 $\quad (\pi.\mathsf{k} = \pi'.\mathsf{k}) \wedge (\pi.\mathsf{peerID} = V) \wedge (\pi'.\mathsf{peerID} = U)$

**Fig. 1:** Correctness experiment for a $k$-message key exchange protocol $\Pi$ between parties $U$ and $V$, with responder's intended peer being $u$.

### 2.2 Security experiment

In experiment $\mathbf{Exp}^{\mathsf{SecPred},F}_{\Pi,n}(\mathcal{A})$, shown in Figure 2, we specify a common execution model that is parametrized by a security predicate SecPred which we later use to capture the different security properties a key exchange protocol might have. In addition, experiment $\mathbf{Exp}^{\mathsf{SecPred},F}_{\Pi,n}(\mathcal{A})$ is parametrized on the protocol $\Pi$, the number of parties $n$ to run in the experiment, and freshness condition $F$.

The session-key indistinguishability property is built into the experiment via the Test query; but other properties can be considered as well. For instance, later in this section we show the less-often-stated property of session key confinement (Confined), i.e., a session key should be shared among at most two sessions. In subsequent sections, we provide additional security predicates to prove our falsification theorem (the soundness and inverse soundness properties in Section 3) and predicates capturing authentication properties in Appendix B.

The freshness condition $F$ models different attacker capabilities (such as forward secrecy); see Section 2.3.

Our approach of encoding each security goal explicitly in its own predicate is different from the modern approach of encoding all goals implicitly through the key indistinguishability property. Our approach yields a modular security model, in which different predicates can be used for protocols with different goals.

[9] We do not assume a global clock; this denotes the local order within $U$'s session.

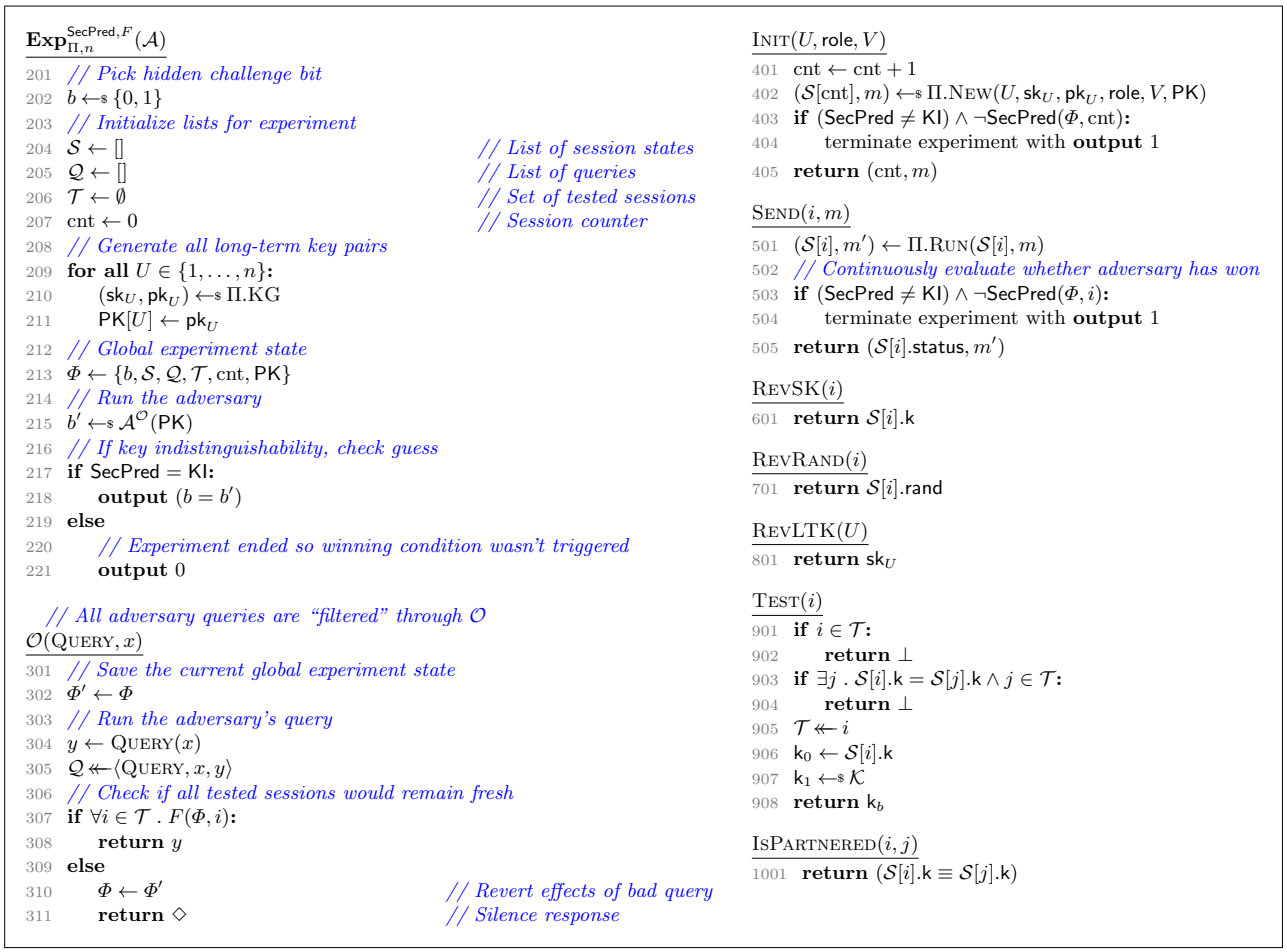


**Fig. 2:** Generic key exchange security experiment for protocol $\Pi$ with $n$ parties against an adversary $\mathcal{A}$, for security property specified by predicate SecPred with freshness condition $F$.

*Experiment overview.* Lines 201–211 of Figure 2 initialize the experiment, which includes: picking a random challenge bit $b$ for the session key indistinguishability game; setting up lists to record session states ($\mathcal{S}$), the adversary's queries ($\mathcal{Q}$), and the sessions that have been tested ($\mathcal{T}$); and generating long-term key pairs for all users. The global experiment state consists of all of those values, and is represented as $\Phi$. Line 215 runs the adversary, who is given all public keys as input, has access to a single oracle $\mathcal{O}$ through which all queries are made, and finally outputs a guess bit $b'$. For the session-key indistinguishability security property, the adversary's guess of the hidden challenge bit is checked on line 218. For the other security properties, the adversary's success in breaking the security property is checked throughout the experiment, specifically on line 503 of the SEND query.

For KI, which is a distinguishing property, we want to bound

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2}\right|.$$

For our remaining properties SecPred, which are win/lose, we want to bound $\Pr\left[\mathbf{Exp}^{\mathsf{SecPred},F}_{\Pi,n}(\mathcal{A}) \Rightarrow 1\right]$. For the latter, we will sometimes write $\Pr[\overline{\mathsf{SecPred}}]$, when $F$, $\Pi$, $n$, and $\mathcal{A}$ are clear from the context.

*Oracle and queries.* Our model provides queries that model an adversary's ability to control all network communications, as well as compromise certain secrets.

The following two queries model normal protocol operation. The adversary uses the INIT query to direct a party $U$ to start a new session with a given role and optional intended peer identifier. The adversary uses the SEND query to deliver a message to a session. Due to the genericity of our experiment, we decided that the INIT and SEND queries *continuously evaluate* the winning condition SecPred every time they are called, and the experiment terminates immediately once the condition is met. This avoids some problems that would develop if the winning condition is not *monotonic*, i.e., if it was possible for a session to enter the winning state, then leave the winning state by the end of the game. (Session-key indistinguishability is still evaluated at the end, since we must wait for the adversary's guess.)

The REVSK, REVRAND, and REVLTK queries model the adversary's ability to learn the session key or randomness, or a party's long-term key. Some AKE security models also allow the registration of malicious public keys (e.g., [16]), but we omit that from our model for simplicity.

The TEST query models the session key indistinguishability security property. As long as the adversary has not already tested this session or its partner (if any), we give the adversary either the real session key or a randomly chosen value. Note that the same hidden challenge bit $b$ is used for every TEST query, so either all TEST queries return real keys, or all TEST queries return random values; Section 5 gives the rationale behind using real-or-random with a single bit across all TEST queries.

The ISPARTNERED query permits the adversary to check whether two sessions are partnered, i.e., have computed the same session key. This enables composability for protocols without public partnering; see Section 4.

However, the adversary is not allowed to query any of these oracles directly. Instead, all queries go through the oracle $\mathcal{O}$. This follows the "silencing" approach of Rogaway and Zhang [60], where the adversary only learns the output of a query if it does not cause tested sessions to become unfresh. E.g., if the adversary queries REVSK for a tested session, this would be a "trivial win" because it would allow the adversary to immediately determine the hidden challenge bit $b$. If $\mathcal{O}$ silences the query, a special silence symbol $\diamond$ is returned, and any changes to the game state are undone. Section 6 discusses alternatives: quantifying over adversaries that never violate freshness, or "penalizing" such adversaries by artificially recording a "loss".

### 2.3 Freshness

Our model is parameterized by a freshness condition $F$, which is used to capture security against different attacker capabilities, such as forward secrecy or the different permitted reveal patterns allowed in the CK [24] or eCK [52] models. Localizing the different attacker capabilities into a parameterized freshness condition follows the approach of Boyd, Cremers, Feltz, Paterson, Poettering, and Stebila [16] and permits comparing the relative strength of security models solely by comparing their freshness conditions. For more discussion, we refer the reader to Section 6.

A freshness condition $F$ with input $(\Phi, i)$ checks whether a particular session $\mathcal{S}[i]$ is fresh based on the global experiment state $\Phi$, which includes all current session states, the list of all non-filtered queries, and the list of tested sessions.

For example, Figure 3 shows a freshness condition ($F^{\mathsf{eCK}}$) capturing the core attacker capabilities of the extended Canetti–Krawczyk (eCK) model [52]. In the eCK model, a session is considered fresh as long as all of the following are satisfied:

1. the session's session key has not been revealed;

```
F^eCK(Φ, i)
1101  // The session's session key has not been revealed
1102  if ⟨RevSK, i, ∗⟩ ∈ Q:
1103      return false
1104  // At most one of the session's ephemeral randomness or
1105  // the owner's long-term key has been revealed
1106  if ⟨RevLTK, S[i].owner, ∗⟩ ∈ Q and ⟨RevRand, i, ∗⟩ ∈ Q:
1107      return false
1108
1109  // For all partner sessions
1110  for all j ≠ i . S[i].k ≡ S[j].k:
1111      // The partner's session key has not been revealed
1112      if ⟨RevSK, j, ∗⟩ ∈ Q:
1113          return false
1114      // At most one of the partner's ephemeral randomness or
1115      // the peer's long-term key has been revealed
1116      if ⟨RevLTK, S[j].owner, ∗⟩ ∈ Q and ⟨RevRand, j, ∗⟩ ∈ Q:
1117          return false
1118
1119  // If there is no partner session, the peer's long-term key
1120  // has not been revealed
1121  if ∄j ≠ i . S[i].k = S[j].k and ⟨RevLTK, S[i].peerID, ∗⟩ ∈ Q:
1122      return false
1123
1124  return true
```

**Fig. 3:** Freshness conditions capturing attacker capabilities similar to the eCK security model [52]. Recall that $\equiv$ treats two values as equal only if they have previously been defined, see notation in Section 2.

2. both the session's ephemeral randomness and the session's owner's long-term secret key have not been revealed (but revealing one or the other is okay);
3. for all partner sessions that exist, we have that both:
   (a) the partner session's session key has not been revealed;
   (b) both the partner session's ephemeral randomness and the peer's long-term secret key have not been revealed (revealing one or the other is okay); and
4. if no partner sessions exist, the peer's long-term secret key has not been revealed.

Different freshness conditions can be used to capture different attacker capabilities, e.g., prohibiting any RevRand query to capture the BR93/BWM model [10,15], or prohibiting revealing the peer's long-term key before acceptance to capture forward secrecy. Figure 4 shows example freshness conditions for attacker capabilities in the BR93/BWM model and the eCK-PFS model [32].[10]
In the remainder of this section, we define two core security properties in our model.

### 2.4 Session key indistinguishability

The first security property that we define using our experiment is session key indistinguishability. As already mentioned, this property is often considered the most central security goal for key exchange protocols. An adversary is deemed to have broken session key indistinguishability if it can distinguish real session keys from random; this is captured in the adversary's ability to guess the hidden challenge bit $b$. We model this in the security experiment by checking if the security predicate is equal to the distinguished symbol $\mathsf{KI}$, which leads to several special cases in the experiment. This allows us to define key indistinguishability as follows:

**Definition 3.** *For a freshness condition $F$ and number of parties $n \in \mathbb{N}$, a protocol $\Pi$ provides $\epsilon$-key-*indistinguishability *against an adversary $\mathcal{A}$ if*

$$\mathsf{Adv}^{\mathsf{KI},F}_{\Pi,n}(\mathcal{A}) := \left| \Pr\left[\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right| \leq \epsilon \ .$$

[10] We do not claim that our models with the corresponding freshness conditions are *equivalent* to the original security models from the literature. For example, BR93 and BWM use matching conversations for partnering, rather than key partnering. Our intention is to represent the permitted query patterns that capture attacker capabilities at a high level.

$F^{\mathsf{BWM}}(\Phi, i)$

```
// The session's session key has not been revealed
if ⟨RevSK, i, ∗⟩ ∈ Q:
    return false
// For all partner sessions
for all j ≠ i . S[i].k ≡ S[j].k:
    // The partner's session key has not been revealed
    if ⟨RevSK, j, ∗⟩ ∈ Q:
        return false
// Neither party's long-term key was revealed
if ⟨RevLTK, S[i].owner, ∗⟩ ∈ Q ∨ ⟨RevLTK, S[i].peerID, ∗⟩ ∈ Q:
    return false
// No ephemeral randomness revealed anywhere
if ⟨RevRand, ∗, ∗⟩ ∈ Q:
    return false

return true
```

$F^{\mathsf{eCK\text{-}PFS}}(\Phi, i)$

```
//
// same as Lines 1101 to 1117 of F^eCK
//
// If there is no partner session, the peer's long-term key has not been revealed before the session accepted
if ∄j ≠ i . S[i].k = S[j].k and ∃r < s . Q[r] = ⟨RevLTK, S[i].peerID, ∗⟩ and Q[s] = ⟨Send, (i, ∗), (accepted, ∗)⟩:
    return false

return true
```

**Fig. 4:** Freshness conditions capturing attacker capabilities similar to the Blake-Wilson–Menezes model [15] (the public key analog of BR93 [10]) and the eCK model with forward secrecy ("eCK-PFS" in [32]).

### 2.5 Session key confinement

Our second security property, *session key confinement*, models the common expectation of two-party key exchange that a particular session key ends up in at most two different sessions. We can capture this either implicitly through key-indistinguishability or explicitly as its own security goal.

In the implicit approach, the adversary is supposed to be able to capitalize on the event that more than two sessions share the same session key by distinguishing the challenge key. That is, once three sessions end up with the same key, they are by definition not considered partners anymore, so the adversary can reveal the session key of one of them and use it to break any of the other two.

While this is a valid encoding of session key confinement, we prefer to state security properties explicitly and thus reward the adversary directly if it manages to get more than two sessions to agree on the same key. Thus, we define session key confinement via the event:

$$\mathsf{Confined}(\Phi, i) : \left| \{ j \mid \Phi.\mathcal{S}[j].\mathsf{k} \equiv \Phi.\mathcal{S}[i].\mathsf{k} \} \right| \leq 2. \tag{1}$$

**Definition 4 (Session key confinement).** *For a freshness condition $F$ and number of parties $n \in \mathbb{N}$, a protocol $\Pi$ provides* $\epsilon$-(session key) confinement *against adversary $\mathcal{A}$ if*

$$\mathsf{Adv}^{\mathsf{Confined},F}_{\Pi,n}(\mathcal{A}) := \Pr\left[\mathbf{Exp}^{\mathsf{Confined},F}_{\Pi,n}(\mathcal{A}) \Rightarrow 1\right] \leq \epsilon \ .$$

When using key-partnering, key-indistinguishability does *not* imply confinement. For example, consider the non-interactive key exchange protocols of Freire, Hofheinz, Kiltz, and Paterson [40], where there may be several sessions between the same pair of parties, each of which is established non-interactively using the same long-term keys and thus leads to the same session key every time. Such protocols provide session key indistinguishability under key partnering (since none of the sessions sharing the same session key can be revealed), but clearly violate confinement.

Note that Equation (1) does not require sessions to be fresh: the adversary may reveal *all* secrets in the experiment. This might seem to make confinement very difficult to achieve. However, for protocols that derive their session keys from a key derivation function (KDF), confinement can usually be proven either by the random oracle assumption, or in the standard model, by assuming collision resistance (satisfied by, e.g., HKDF [49]).

## 3 Falsifiability and partnering

As noted in the introduction, key exchange security models using existential partnering [3, 9, 11, 19, 21–25, 34, 38, 46, 50, 59, 61] allow the prover to state a session identifier or partner function for which their protocol can be proven secure, rather than the model providing one. We call this a *partnering mechanism*: the security model explicitly defines a relation that decides whether two sessions should be considered partners or not.

Without further restrictions on this relation, it is possible to define unnatural and pathological mechanisms that allow intuitively insecure protocols to be proven secure, or mechanisms that make *all* protocols insecure.

For example, a partnering mechanism that partners *all* sessions artificially limits the adversary's powers, since it cannot reveal the session key of *any* session. As a result, protocols where the session key of different sessions are not independent of each other can be proven secure. More generally, allowing a partnering mechanism that partners everyone—even sessions with different session keys—is an example of *over-provisioning*, since it partners sessions that intuitively should have nothing to do with each other.

At the other end of the spectrum is a partnering mechanism which partners *no one*. This is an example of partner *under-provisioning* since it allows attacks in the model that do not correspond to any real-world attacks.

In this section we formalize soundness and "inverse soundness" that capture over- and under-provisioning respectively. We then show that a protocol that is secure with respect to a partnering mechanism that does not over- or under-provision is also secure with respect to key partnering; we call this the baseline theorem of key exchange partnering.

### 3.1 Partnering

**Definition 5 (Partnering mechanism).** *Let $\mathcal{I}$ be the space of all instance states. A* partnering mechanism *is a binary relation on $\mathcal{I}$.*

For example, key partnering is $P_{\text{key}}(\pi, \pi') = (\pi.\mathsf{k} = \pi'.\mathsf{k})$; matching conversations is

$$\begin{aligned} P_{\text{mc}}(\pi, \pi') = (&\pi.\mathsf{transcript} = \pi'.\mathsf{transcript}) \\ &\vee (\exists m \,.\, \pi.\mathsf{transcript} = \pi'.\mathsf{transcript} \| [m])) \\ &\vee (\exists m \,.\, \pi.\mathsf{transcript} \| [m] = \pi'.\mathsf{transcript})) \end{aligned}$$

Our security experiment and freshness conditions in Section 2 are stated with key-partnering already built into the definitions. For the purposes of this section, we need to generalize them to an arbitrary partnering mechanism $P$, which is done simply by replacing all session key equality checks with the general partnering check.

In particular, we define $\mathbf{Exp}^{\mathsf{SecPred},F,P}_{\Pi,n}$ by making the following modifications in Figure 2:

- Test line 903 becomes: "if $\exists j \,.\, P(\mathcal{S}[i], \mathcal{S}[j])$ and $j \in \mathcal{T}$:"
- IsPartnered line 1001 becomes: "return $P(\mathcal{S}[i], \mathcal{S}[j])$"

The freshness condition $F$ is also allowed to depend on the partnering mechanism $P$. For example, in Figure 3:

- $F^{\mathsf{eCK}}$ line 1110 becomes: "for all $j \,.\, P(\mathcal{S}[i], \mathcal{S}[j])$:"
- $F^{\mathsf{eCK}}$ line 1121 becomes: "if $\nexists j \,.\, P(\mathcal{S}[i], \mathcal{S}[j])$ and $\exists \ldots$"

Note that our partnering mechanism compares session states, so our security experiment uses indices $i, j$, etc. to index into the list of sessions $\mathcal{S}$ and then evaluates the partnering mechanism on session states $\mathcal{S}[i], \mathcal{S}[j]$.

We now turn to assessing whether a partnering mechanism over- or under-provisions session partners, which we will model by certain soundness properties.[11]

Beginning with the problem of over-provisioning, we demand that partners should derive the same session key. This is captured by the following event defined on security experiment $\mathbf{Exp}^{\mathsf{SecPred},F,P}_{\Pi,n}$:

$$\mathsf{Sound}_P(\Phi, i) \colon \forall \pi, \pi' \in \Phi.\mathcal{S} \,.\, P(\pi, \pi') \implies \pi.\mathsf{k} = \pi'.\mathsf{k}.$$

$\epsilon$-soundness is defined analogously to Definition 4.

To deal with the issue of under-provisioning, we demand that any two sessions that derive the same session key should also be partners. We call this *inverse soundness*, defined by the event:

$$\mathsf{InvSound}_P(\Phi, i) \colon \forall \pi, \pi' \in \Phi.\mathcal{S} \,.\, \pi.\mathsf{k} \equiv \pi'.\mathsf{k} \implies P(\pi, \pi').$$

$\epsilon$-inverse-soundness is defined analogously to Definition 4.

Notice both soundness and inverse soundness are required to hold unconditionally with respect to session freshness: each must hold even when the adversary can obtain any secret value it wants.

Soundness is one of the conditions required of Match security [22]. Inverse soundness is seldom mentioned in key exchange models, but was described by Kudla and Paterson [51] as *strong partnering*. Together, these two properties allow us to prove in the next section our baseline theorem relating security under key partnering to security under arbitrary partnering mechanisms.

[11] Here, "soundness" refers to a property of the partnering mechanism; we use the term "correctness" for the property that honest parties, in the absence of active adversarial interference, derive equal session keys.

### 3.2 Baseline theorem of key exchange partnering

**Theorem 1 (Baseline theorem of key exchange partnering).** *Let* $\Pi$ *be a key exchange protocol. For any security property* $\mathsf{SecPred}$*,* $\Pi$ *is secure under key-partnering if and only if it is secure under* $P$*-partnering, as long as the partnering mechanism* $P$ *is sound and inverse-sound. More precisely, for all* $\mathsf{SecPred}$*,* $\Pi$*,* $n$*,* $F$*,* $P$*, and* $\mathcal{A}$*,*

$$\left|\mathsf{Adv}^{\mathsf{SecPred},F,P_{\text{key}}}_{\Pi,n}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{SecPred},F,P}_{\Pi,n}(\mathcal{A})\right| \leq \Pr[\overline{\mathsf{Sound}}_P] + \Pr[\overline{\mathsf{InvSound}}_P]. \tag{2}$$

Note that while the same $F$ is used in the two experiments in (2), that $F$ may call the partnering mechanism used in the respective experiment, so we would have "$F$-with-$P_{\text{key}}$" or "$F$-with-$P$". Also recall that we use $\Pr[\overline{\mathsf{Sound}}_P]$ and $\Pr[\overline{\mathsf{InvSound}}_P]$ as a short-hand for the advantage an adversary has in breaking soundness or inverse-soundness of $\Pi$ with $P$. Interestingly, de Saint Guilhem, Fischlin and Warinschi [33, Theorem 5.1] prove that if Match security holds, then equal keys implies equal partners already so that the requirement of inverse soundness might seem superfluous. However, their implication only holds for *fresh* sessions and thus, the additional requirement of inverse soundness is needed in our theorem.

A direct consequence of Theorem 1 is the falsifiability of security models using session identifiers or general partnering functions. If an attack is shown against a security property of a key exchange protocol when using a sound and inverse-sound partnering mechanism, then that is indeed an attack against the protocol under key partnering or (by transitivity) under any other sound or inverse-sound partnering mechanism.

*Proof.* Consider the run of $\mathbf{Exp}^{\mathsf{SecPred},F,P}_{\Pi,n}(\mathcal{A})$ with the same random coins for the experiment and the adversary as in the run of $\mathbf{Exp}^{\mathsf{SecPred},F,P_{\text{key}}}_{\Pi,n}(\mathcal{A})$, but using $P$ instead of $P_{\text{key}}$. Let $\mathsf{Same}$ be the event that $P(\pi,\pi') = P_{\text{key}}(\pi,\pi')$ at every evaluation of the partnering mechanism in the experiment and freshness conditions, as described earlier in Section 3.1; $\overline{\mathsf{Same}}$ is the complement of $\mathsf{Same}$. Then the two runs behave identically as long as $\overline{\mathsf{Same}}$ does not occur, i.e.,

$$\left|\mathsf{Adv}^{\mathsf{SecPred},F,P_{\text{key}}}_{\Pi,n}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{SecPred},F,P}_{\Pi,n}(\mathcal{A})\right| \leq \Pr[\overline{\mathsf{Same}}]. \tag{3}$$

If $\overline{\mathsf{Same}}$ occurs, and thus there is some point in time for which there is some pair of sessions $\pi,\pi'$ for which $P(\pi,\pi') \neq P_{\text{key}}(\pi,\pi')$, then either (a) $P(\pi,\pi')$ but $\pi.\mathsf{k} \neq \pi'.\mathsf{k}$ (which we will show violates soundness for $P$), or (b) $\pi.\mathsf{k} = \pi'.\mathsf{k}$ but $\neg P(\pi,\pi')$ (which will violate inverse-soundness for $P$).

There are three places within the experiment which can cause the event $\overline{\mathsf{Same}}$ to occur, namely the three places where we modified $\mathbf{Exp}^{\mathsf{SecPred},F}_{\Pi,n}$ to $\mathbf{Exp}^{\mathsf{SecPred},F,P}_{\Pi,n}$ at the start of Section 3.1:

- Line 903 of the $\textsc{Test}(i)$ query: If there is some $j$ for which $P(\mathcal{S}[i],\mathcal{S}[j]) \neq P_{\text{key}}(\mathcal{S}[i],\mathcal{S}[j])$, then this would also have been true at the most recent $\textsc{Init}$ or $\textsc{Send}$ query involving either $\mathcal{S}[i]$ or $\mathcal{S}[j]$. Note that we only have to consider $\textsc{Init}$ and $\textsc{Send}$ queries since they are the only queries that modify session variables, and we only have to consider the most recent such query involving one of those sessions since $\textsc{Init}$ or $\textsc{Send}$ queries to other sessions do not affect the partnering of $\mathcal{S}[i]$ or $\mathcal{S}[j]$.
- Line 1001 of the $\textsc{IsPartnered}$ query: Similarly.
- Inside the call to $F$ on line 307 of the $\mathcal{O}$ oracle: The freshness condition $F$ may evaluate the partner predicate zero or more times, on two arbitrary sessions $\pi,\pi' \in \mathcal{S}$. Note that $F$ uses the partnering mechanism of its experiment, therefore it is either $F$-with-$P_{\text{key}}$ or $F$-with-$P$, depending on whether we are in $\mathbf{Exp}^{\mathsf{SecPred},F,P_{\text{key}}}_{\Pi,n}(\mathcal{A})$ or $\mathbf{Exp}^{\mathsf{SecPred},F,P}_{\Pi,n}(\mathcal{A})$. If, at any of $F$'s evaluations of the partnering mechanism, we have that $P(\pi,\pi') \neq P_{\text{key}}(\pi,\pi')$, then it would also have been true at the most recent $\textsc{Init}$ or $\textsc{Send}$ query involving either $\pi$ or $\pi'$.

Thus, if $\overline{\mathsf{Same}}$ occurs in $\mathbf{Exp}^{\mathsf{SecPred},F,P}_{\Pi,n}(\mathcal{A})$, then either the game $\mathbf{Exp}^{\mathsf{Sound},F,P}_{\Pi,n}(\mathcal{A})$ or the experiment $\mathbf{Exp}^{\mathsf{InvSound},F,P}_{\Pi,n}(\mathcal{A})$ (with the same random coins for the experiment and adversary) outputs 1. Hence

$$\Pr[\overline{\mathsf{Same}}] \leq \Pr[\overline{\mathsf{Sound}}_P] + \Pr[\overline{\mathsf{InvSound}}_P]. \ .$$

Combining (3) with the above inequality yields the result.

Note that key partnering is clearly perfectly sound and inverse-sound.

At first glance, Theorem 1 might seem vacuous: security with $P$-partnering approximates security with key-partnering if $P$-partnering approximates $k$-partnering. However, there are variants of $\mathbf{Exp}^{\mathsf{SecPred},F}_{\Pi,n}$ for which proving the baseline theorem becomes unclear. For example, we initially tried to write $\mathbf{Exp}^{\mathsf{SecPred},F}_{\Pi,n}$ with all security predicates evaluated at the end of the main experiment on line 218, rather than continuously evaluating non-$\mathsf{KI}$ predicates in the $\textsc{Send}$ query as Figure 2 shows. We were unable to prove the corresponding baseline theorem: soundness/inverse-soundness would only be guaranteed at the end of the experiment, but there might have been intermediate points where it was temporarily violated, which might result in different behavior between the experiment using key-partnering versus $P$-partnering.

Theorem 1 uses a generic security predicate: it holds for, e.g., session-key indistinguishability, confinement, and the authentication properties in Appendix B.

## 4 Composition should be possible

Brzuska, Fischlin, Warinschi, and Williams [22] (BFWW) show that if a key exchange protocol is composable, then it is possible to (weakly) determine which sessions derive the same keys only based on the public *protocol* transcript. However, BFWW consider a model that does not expose a session matching oracle, and we argue now, that if the *model* itself exposes a session matching oracle, then the key exchange protocol can actually be composable without admitting a public session-matching algorithm based on transcripts only. That is, we show that the class of key exchange protocols that are securely composable is bigger than the class identified by BFWW. In this section, we first explain why a key exchange secure in security model with a session matching oracle is composable and then discuss a separating example of a key exchange protocol that is composable, intuitively and provably, but was excluded by BFWW due to the absence of a public session-matching algorithm.

### 4.1 Composability

In order to prove that a key exchange model provides composability with a symmetric-key primitive, one first needs a definition of security for the symmetric-key primitive and a definition of a composed game. Let's think of $k \leftarrow\!\!\$\ \{0,1\}^n$ as being a line of pseudo-code in the game defining the security of the symmetric primitive. The composed game will replace this line by using the session key of the key exchange game. Besides, the composed game will expose the same queries to the adversary as the game defining the security of the symmetric primitive and the key exchange game, except for Test and Reveal queries. The composed game uses bit $b = 0$ for the key exchange, and the adversary wins the composed game based on the winning condition of the symmetric primitive.

To reduce the composed security to the two underlying building blocks, one first reduces to the key exchange security to replace real session keys with random session keys. Then, one can reduce to the security of the symmetric-key protocol. Making this proof outline rigorous is less straightforward than one might think. A tricky part in the proof is that the symmetric primitive game needs to be multi-session and key exchange sessions that belong together must be mapped to the same instance of the symmetric primitive. Therefore, in the reduction to the key exchange, the reduction needs to know which two sessions are partnered. BFWW [22] thus argued that a protocol must have a public matching algorithm. This approach was also followed by [20] for composition of non-forward secure key exchange protocols and by Skrobot and Lancrenon [62] for password-based authenticated key exchange. Moreover, due to the session identifiers in Universal Composability, also Canetti and Krawczyk [26] assumed the protocol to have public partnering. In turn, Brzuska, Delignat-Levaud, Fournet, Kohbrok, and Kohlweiss [18] and George and Rackoff [42] provide the adversary with a session-matching oracle that tells the adversary which pairs of sessions are partnered. In this paper, we argue for the advantages of the latter choice. Namely, it allows to establish secure composability of a larger class of protocols.

### 4.2 A separating example

Let $\Pi$ be a key exchange protocol that is secure in an arbitrary key exchange model with mutual authentication and pre-specified peers. We now add public-keys for a re-randomizable encryption scheme to $\Pi$ and encrypt all messages of the original $\Pi$ protocol with re-randomizable encryption of the intended peer. (Here, we use mutual authentication and pre-specified peers). We obtain a new protocol $\Pi'$. In the previous subsection, we showed that protocols secure in a model with a partnering oracle are composable. In this section, we show that $\Pi'$ is indeed secure in a model with a partnering oracle but $\Pi'$ does not have a public partnering mechanism.

$\Pi'$ *is secure in a model with a partnering oracle.* Let $\mathcal{A}$ be an adversary against $\Pi'$ in a model with a partnering oracle. We now build an adversary $\mathcal{B}$ against $\Pi$. $\mathcal{B}$ first draws all keys for the rerandomizable encryption scheme and whenever a party $P_i$ with intended peer $P_j$ sends a message $m$, then $\mathcal{B}$ encrypts $m$ under the public key of $P_j$ with the rerandomizable encryption scheme. In turn, when the adversary makes a $\textsc{Send}(i, m)$ query to session $i$, and $P$ is the owner of session $i$, then $\mathcal{B}$ first decrypts $m$ using the secret key of $P$ of the rerandomizable encryption scheme and forwards the decrypted message to the experiment. All other oracle queries are forwarded. The soundness of the simulation is a bit hard to argue in an arbitrary model, but the emulation of the Send query is perfect, RevSK, IsPartnered, RevLTK (here, we need to add the secret key) also return the same answer, RevRand (here, we need to add the randomness for the rerandomizable encryption scheme).

$\Pi'$ *does not have public partnering.* Consider an adversary $\mathcal{A}$ that creates two sessions for $P_i$ and two sessions for $P_j$, flips a bit to see which one is matched to which and then re-randomizes messages. By security of rerandomizable encryption, from the public transcript, one cannot tell which session is matched with which session. If one wants these probabilities to be more dramatic, one can take many pairs of such sessions and gets a guessing probability of $\frac{1}{2}^c$, where $c$ is the number of sessions: the guessing probability is upper bounded by $\frac{1}{2}^c$ or the probability of breaking the rerandomizable encryption scheme. If $c$ is polynomial, the guessing probability is negligible.

### 4.3 A general composition theorem

We can bypass the aforementioned counterexample and impossibility result by Brzuska, Fischlin, Warinschi and Williams [22], as we added a partnering oracle to our model which tells the adversary whether two sessions are partnered or not. A similar observation was made by Brzuska, Delignat-Lavaud, Fournet, Kohbrok and Kohlweiss [18] who establish composability of a specific eCK variant. We can generalize their theorem to arbitrary key exchange protocols which can be formalized in our above model, regardless of their freshness predicate. To prove such a general composability theorem, we need to formalize the above mechanism that defines the composition of a key exchange protocol with a symmetric-key primitive. To be able to do this, we use the technique of [18] to slice code into several pieces of code. The first object we need is a keys array which will replace some of the code of the Test oracle.

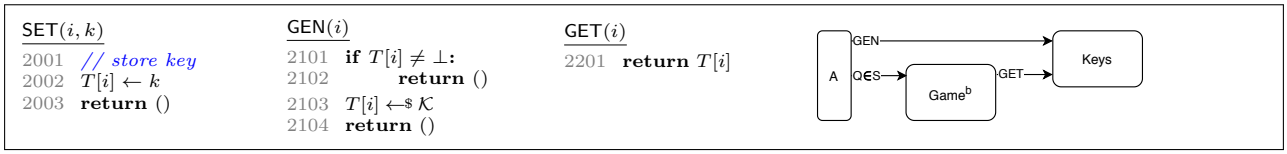


**Fig. 5:** Keys Array

**Definition 6 (Keys Array).** *A* Keys *array is a piece of pseudocode which exposes the oracles* SET, GEN *and* GET *which behave as specified in Figure 5.*

Now, we first define a symmetric-key security game which relies on the Keys array and then modify our key exchange experiment to interact with the Keys array, too.

**Definition 7 (Symmetric-Key Security Game).** *Let* $\mathsf{G}^0$ *and* $\mathsf{G}^1$ *be stateful pieces of pseudocode that expose the same set* $S$ *of oracles to the adversary and make queries to the* GET *oracle of a* Keys *array. Then, we define the game* $\mathsf{G}^b \to \mathsf{Keys}$ *as the game where an adversary can call oracles* $S \cup \{\mathsf{GEN}\}$*, where* GEN *calls of the adversary are executed by* Keys*.* $\mathsf{G}^b \to \mathsf{Keys}$ *is depicted on the right side of Figure 5. For an adversary* $\mathcal{A}$ *interacting with* $\mathsf{G}^b \to \mathsf{Keys}$*, we define the advantage as*

$$\epsilon_{\mathsf{G}\to\mathsf{Keys}}(\mathcal{A}) := \left|\Pr[1 = \mathcal{A} \to \mathsf{G}^0 \to \mathsf{Keys}] - \Pr[1 = \mathcal{A} \to \mathsf{G}^1 \to \mathsf{Keys}]\right|$$

The key exchange experiment $\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}(\mathcal{A})$ does not terminate early and it always terminates in line 215. Additionally, it does not rewind the adversary. We can thus *externalize* the adversary and write $\mathcal{A} \to \mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}$ as an adversary which interacts with the oracles of $\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}$. Additionally, we can fix the bit $b$ in $\mathbf{Exp}^{\mathsf{KI},F,b}_{\Pi,n}$ and change the Test query such that, instead of returning a key to the adversary, it writes the key into Keys via a $\mathsf{SET}(i,k)$ query if $b = 0$ and makes a $\mathsf{GEN}(i)$ query to Keys if $b = 1$. The adversary is now given access to the GET oracle of Keys.

**Definition 8 (Composable Key Exchange Game).** *Let* $\mathcal{A} \to \mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}$ *be an adversary that interacts with the oracles of* $\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}$*, where we fix* $b$ *in* $\mathbf{Exp}^{\mathsf{KI},F,b}_{\Pi,n}$*. Let the Test query write the key into* Keys *via a* $\mathsf{SET}(i,k)$ *query if* $b = 0$ *and make a* $\mathsf{GEN}(i)$ *query to* Keys *if* $b = 1$*. The adversary is given access to the* GET *oracle of* Keys*. We define the adversary's external advantage as* $\epsilon_{\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}\to\mathsf{Keys}}(\mathcal{A}) :=$

$$\left|\Pr[1 = \mathcal{A} \to \mathbf{Exp}^{\mathsf{KI},F,0}_{\Pi,n} \to \mathsf{Keys}] - \Pr[1 = \mathcal{A} \to \mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n} \to \mathsf{Keys}]\right|.$$

Note that $\epsilon_{\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}\to\mathsf{Keys}}(\mathcal{A})$ is twice the standard advantage. We can now naturally define the composed game where $\mathbf{Exp}^{\mathsf{KI},F,b}_{\Pi,n}$ is connected to Keys via a SET query (if $b = 0$) or via a GEN query (if $b = 1$), $\mathsf{G}^{b'}$ is connected to Keys via a GET query, and the adversary has access to the oracles of $\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}$ and $\mathsf{G}$ (but not to any oracle of Keys). In line with [18], we denote the parallel composition of two games by a fraction notation.

**Definition 9 (Composed Game).** *Let* $\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}(\mathcal{A})$ *be a key exchange game from our family, let* $\mathsf{G}^0$ *and* $\mathsf{G}^1$ *be a symmetric-key security game, then we define the advantage of* $\mathcal{A}$ *against their composition as* $\epsilon_{comp}(\mathcal{A}) :=$

$$\left|\Pr[1 = \mathcal{A} \to \frac{\mathbf{Exp}^{\mathsf{KI},F,0}_{\Pi,n}}{G^0} \to \mathsf{Keys}] - \Pr[1 = \mathcal{A} \to \frac{\mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n}}{G^1} \to \mathsf{Keys}]\right|.$$

**Theorem 2.** *Let* $\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}(\mathcal{A})$ *be a key exchange game from our family, let* $\mathsf{G}^0$ *and* $\mathsf{G}^1$ *be a symmetric-key security game, then*

$$\epsilon_{comp}(\mathcal{A}) \leq \epsilon_{\mathbf{Exp}^{\mathsf{KI},F}_{\Pi,n}\to\mathsf{Keys}}(\mathcal{A}\to\mathsf{G}^0) + \epsilon_{\mathsf{G}\to\mathsf{Keys}}(\mathcal{A}\to\mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n})$$

*Proof.* The proof is purely syntactical. In the first game hop, we move from $\frac{\mathbf{Exp}^{\mathsf{KI},F,0}_{\Pi,n}}{G^0}\to\mathsf{Keys}$ to $\frac{\mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n}}{\mathsf{G}^0}\to\mathsf{Keys}$ by making the code of $\mathsf{G}^0$ part of the adversary, i.e., we consider $\mathcal{A}\to\mathsf{G}^0$ together as an adversary against the key exchange. In the game-hop from $\frac{\mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n}}{\mathsf{G}^0}\to\mathsf{Keys}$ to $\frac{\mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n}}{\mathsf{G}^1}\to\mathsf{Keys}$, we do the converse, i.e., we make the code of $\mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n}$ part of the adversary and consider $\mathcal{A}\to\mathbf{Exp}^{\mathsf{KI},F,1}_{\Pi,n}$ as an adversary against $\mathsf{G}^b$. This concludes the proof.

## 5 Composition should be tight

In the previous section we argued that for an AKE security notion to be useful it should be possible to compose it with other security notions. Here we go one step further and argue that composition should also be efficient. By efficient we mean in the sense of practice-oriented provable security [58]: a reduction from the composed protocol to the AKE should be *tight* [27, 28]. More concretely, suppose you have a protocol $\Pi = \mathsf{KE};\Sigma$ consisting of the composition of an AKE protocol $\mathsf{KE}$ and symmetric protocol $\Sigma$, i.e., where the keys used by $\Sigma$ are generated by $\mathsf{KE}$. Now assume $\mathsf{KE}$ is secure according to some composable AKE security notion $\mathsf{AKE}$, $\Sigma$ is secure according to some notion $X$, and the goal is to show that $\Pi$ is secure according to some notion $Y$. Then we want the $Y$-security of $\Pi$ to be tightly reducible to the $\mathsf{AKE}$-security of $\mathsf{KE}$ and the $X$-security of $\Sigma$, informally stated:

$$\mathsf{Adv}^{Y}_{\mathsf{KE};\Sigma} \leq \mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{KE}} + \mathsf{Adv}^{X}_{\Sigma}. \tag{4}$$

For example, $\Sigma$ could be an authenticated encryption scheme, $X$ could be the security notion of multi-user authenticated encryption (mu-AE) [45], and $Y$ could be the security notion of authenticated and confidential channel establishment (ACCE) [44].

Intuitively, this should be possible since an AKE protocol is fundamentally a multi-user object, and so the security of $\mathsf{KE}$ should "line-up" with the multi-user security of $\Sigma$ to provide security for their composition $\Pi$. In particular, we want the AKE security notion to support the following natural proof strategy: start by replacing the session keys of all fresh sessions with random keys (which can be done since $\mathsf{KE}$ is secure), then appeal to the $X$-security of $\Sigma$ to argue that the composition $\Pi = \mathsf{KE};\Sigma$ now satisfies $Y$-security. This argument has been formalized by Brzuska, Fischlin, Warinschi, and Williams [22], showing that BR-secure AKE protocols can be composed with arbitrary symmetric-key protocols. Unfortunately, the reduction given in [22] is not quite of the form (4), but rather

$$\mathsf{Adv}^{Y}_{\mathsf{KE};\Sigma} \leq q \cdot \mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{KE}} + \mathsf{Adv}^{X}_{\Sigma}, \tag{5}$$

where the factor $q = n_U^2 \cdot n_s$ depends on the number of users $n_U$ and the number of sessions per user $n_s$. For systems with billions of users and sessions such as TLS, the factor $q$ can become very substantial. As a result, if parameters are to be selected in a theoretically sound manner supported by reductions, they would have to be increased significantly, thereby hurting performance.

So where does the factor $q$ in (5) come from? It comes from a hybrid argument in [22] where, one-by-one, the session keys of all fresh sessions are replaced with random keys. The hybrid argument is necessary since the AKE model in [22] only allows one Test query. A 1-Test model is thus not conducive to a tight composition result like (4). More conceptually, we see the 1-Test model as failing to reflect the multi-user nature of key exchange.

*$n$-FtG vs. RoR.* Given that a 1-Test model is inadequate for tight composition, the natural solution is to use an $n$-Test model where the adversary can make multiple Test queries. But there are two reasonable ways in which this can be done: the $n$-FtG (Find-then-Guess) model, where each session is equipped with its own independent secret bit $b_i$ and each Test query is answered real-or-random based on the corresponding session's secret bit; or the RoR (Real-or-Random) model, where all Test queries are either all answered with real keys, or all are answered with random keys, based a *single* secret bit $b$.[12] Both the $n$-FtG model [4, 34, 38, 43] and the RoR model [1–3] have seen use in the literature. So which one should you prefer? Answer: RoR.

The $n$-FtG model is no better than the 1-FtG model when it comes to tight composition, because in the reduction it does not allow replacement of all fresh session keys with random keys in one big swoop due to the secret bits $b_i$ being independent. On the other hand, the RoR model allows all fresh session keys to be replaced at once. Thus, the proof of (4) is simply a matter of "lining up" the keys from $\mathsf{KE}$ with the correct instances of the symmetric protocol $\Sigma$. In fact, Skrobot and Lancrenon [62] have carried out exactly this proof by adapting the composition framework of BFWW [22] to the RoR setting (albeit for password-based protocols).

[12] The FtG and RoR labels are inspired by the similarly-named IND-CPA security notions for symmetric encryption [7].

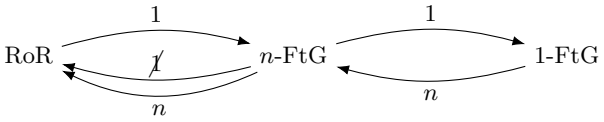


**Fig. 6:** Relationship between notions; $X \xrightarrow{L} Y$ indicates that notion $X$ implies notion $Y$ with security loss $O(L)$.

Comparing the RoR and $n$-FtG models (see Figure 6), one can show that RoR tightly implies $n$-FtG, while $n$-FtG only implies RoR with a tightness loss of $n$. Moreover, this loss is inherent. All of these claims can be proven by adapting the corresponding proofs in [3] to our model given in Section 2. Note that this is not just an exercise in moving definitions around so that a tightness gap is hidden elsewhere. For example, our proof in Appendix A of the security of the NAXOS++ protocol obtains the same tightness gap to the underlying hardness assumptions in the RoR model as the original proof of the NAXOS+ protocol did in the 1-FtG model [54], but due to the RoR-model there is no additional gap when composing the AKE protocol with a subsequent symmetric protocol.

Finally, we note a peculiarity of the $n$-FtG model. For security to be meaningfully defined in the 1-FtG and RoR models all test sessions must be fresh, otherwise the adversary could trivially win the game. However, in the $n$-FtG model—where the adversary's output $(i, b')$ is a guess of the singular session $i$'s secret bit—one could technically allow the compromise of all the *other* test sessions, since this wouldn't necessarily trivialize the game. But we do not recommend this variant of $n$-FtG. First of all, we find it conceptually wrong, since the whole purpose of the Test query is to measure the adversary's ability to distinguish session keys of valid targets. If the adversary really wanted to learn the keys of the other test sessions it should have used the Reveal query. Second, with this variant we can no longer prove the implication RoR $\implies$ $n$-FtG.

*Tight AKE constructions vs. tight AKE composition.* This section has focused on the usefulness of the AKE security notion itself, i.e., how tightly can the security of a complex protocol be reduced to the security of the underlying AKE? In a sense, we have focused on the *user* of the AKE security notion.

In contrast, tightness considerations in the literature have mainly focused on the *construction* of the AKE protocol itself, i.e., how tightly can the security of the AKE protocol be reduced to the security of some underlying building block, such as Diffie-Hellman or RSA? So far, only a few AKE protocols with tight, or nearly tight, reductions are known [4, 5, 29, 43].

Note that these two types of tightness considerations are complementary. For the security proof of the overall system to be maximally meaningful (in the sense of practice-oriented provable security [6, 28, 41, 57, 58]), both the construction and the composition need to be tight.

## 6 Comparability of models

The main reason to compare key exchange security models is to compare the relative *strength* of a considered adversary, i.e., which capabilities are the adversary assumed to have in the model? However, existing models typically entangle the capabilities in slightly different ways, and hence there are no common capability parameters that could serve as a basis for comparison.

If one commits to only using a specific family of models parametrized solely by a freshness condition (such as the one in [32], [16], or our $\mathbf{Exp}^{\mathsf{SecPred},F}_{\Pi,n}$ in Section 2), then the comparison boils down to comparing the freshness conditions. However, in practice, other aspects may also differ, such as the choice of partnering mechanism [31]. Nevertheless, our baseline theorem shows that the behavior of different partnering mechanisms is approximately the same, provided they satisfy the two natural soundness properties. Thus, here we focus instead on another source of incomparability, namely the treatment of adversarial misbehavior, which has not been considered by previous works.

For a given property, we aim to determine a protocol's security against adversaries that do not violate the freshness condition; what we will call *well-behaved* adversaries. Surprisingly, there is no consensus on how to ensure that only well-behaved adversaries are considered in the security definition, as illustrated by the following different approaches taken in the literature.

*A) Exclusion-style.* In this approach one simply quantify over the well-behaved adversaries only. The security experiment is typically formulated as follows (see, e.g., [16]):

1. The experiment begins, and the adversary can issue any permissible query.
2. At some point it issues a Test query to a *fresh* session.
3. It continues issuing queries, *under the condition that the test session remains fresh.*
4. Finally, the adversary outputs a guess $b'$.

While the exclusion-style formulation is probably the one most commonly found in the literature, it has some conceptual drawbacks. Quoting Rogaway and Zhang [60]:

> Exclusion-style definitions compel consideration of adversary classes. They disqualify adversaries that only rarely misbehave. They ignore whether or not an adversary can 'know' it has misbehaved. And they promote ambiguity, as the relevant restrictions are not expressed in game code.

We refer the reader to [60] for further details.

*B) Penalty-style.* Another approach is to quantify over *all* adversarial behaviors, but then penalize the misbehavior at the end of the experiment (e.g, by outputting a random bit on the adversary's behalf). The difference between the exclusion-style and the penalty-style definitions has previously been considered by Bellare, Hofheinz, and Kiltz [8] in the context of IND-CCA security for public-key encryption. Examples of models using the penalty-style are given in [9, 19, 35, 54]. For a proof using a penalty-style definition, the relevant adversary restrictions will manifest themselves during the probability analysis where one needs to check that indeed, the reduction will not be penalized by the game it is playing and/or that it is penalized if and only if the original adversary would have been penalized in its game as well. These analyses can sometimes be quite subtle.

*C) Filtering-style.* Finally, we have the approach we prefer, where queries that constitute adversarial misbehavior are not executed, and no response is returned to the adversary. This filtering-style definition is inspired by George and Rackoff [42] and the work of Rogaway and Zhang [60].[13] The advantage of a filtering-style definition is that it makes the accepted adversarial behaviour explicit in the game code, and it avoids the need for subtle freshness analyses at the end of the proof.

*Relations between notions.* The relationship between the three notions is subtle. First, security in a filter-style model implies security in an exclusion-style model, but not the other way around. The problem is that one cannot always publicly check whether a query is valid or not. However, if the validity can be publicly checked (i.e., if it does not depend on secret game state), then exclusion-style security implies filter-style security. This is similar to a result by Rogaway and Zhang [60].

For penalty-style security the situation is much more complicated. First, similar to the direction exclusion-style security $\rightarrow$ filter-style security, security in a penalty-style model only implies security in a filter-style model if one can publicly check validity. However, in the converse direction filter-style security fails to imply penalty-style security. To illustrate this, consider a penalty-style adversary $\mathcal{A}$ for which we aim to build a filter-style adversary $\mathcal{B}$ (against the same protocol) using the eCK-like freshness predicate $F^{\mathsf{eCK}}$ given in Figure 3. Now suppose $\mathcal{A}$ behaves as follows: (1) it reveals all long-term keys; (2) it forwards messages passively between two sessions until one of them accepts; (3) it tests this session; (4) it delivers the test session's final message to the other one (so they become partners); (5) it stops and outputs a guess. The problem for $\mathcal{B}$ occurs in step (3): at this point the test-session is non-fresh according to $F^{\mathsf{eCK}}$, so it won't get a response back if it forwards $\mathcal{A}$'s Test query to its own filter-style game. However, it can't simply abort, because $\mathcal{A}$ is a valid penalty-style adversary due to step (4) (since the test-session eventually gets a partner, it is fresh by the time of step (5)).

The reason for this issue is that in a penalty-style model the "intermediate" freshness state of a session could be *non-monotonic*. I.e., even though a session is fresh when the experiment ends, it could have been considered unfresh at certain points during the experiment. In our view, this non-monotonic aspect of the penalty-style model is counter-intuitive and complicates reasoning. Also, it is not clear whether the obstacle to proving that filter-style security implies penalty-style security represents an actual security difference, or whether it is merely a proof technicality.

## 7 Discussion

For the cryptographer developing a protocol, we offer a family of key exchange models in Section 2, parameterized by a freshness condition tailored to capture the intended adversarial attack capabilities. A proof in one of our models ensures that no attack exists under a different reasonable partnering mechanism, and that efficient composition with a symmetric-key protocol is possible.

Our results are useful beyond the family of models: for those who prefer to use session identifiers rather than key partnering for your proof, our baseline theorem of key exchange partnering says this is fine, as long as soundness of the session identifiers is proven. For those who prefer to penalize adversaries that violate the freshness condition, rather than filtering the response from unfresh queries, the IsPartnered oracle provides the public checkability of partnering to show these equivalent.

Our results show that by some careful choices for key exchange models, one can relatively easily obtain sanity in interpreting and relating different key exchange security models, and assurance that protocols satisfying those models can be composed in reasonable ways.

[13] In the *silencing* definition of Rogaway and Zhang [60], the game state is updated and only the response is suppressed, whereas our formulation in Figure 2 reverts the game state if the response is to be suppressed.

**Acknowledgements**

We thank Konrad Kohbrok for discussions on the baseline theorem of key partnering at early stages of this work. We thank Eric Cornelissen for helpful comments on the presentation.

D.S. is supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant RGPIN-2022-03187. This work was supported by the Research Council of Finland.

## References

1. Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press. 16
2. Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 332–352, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Berlin, Heidelberg, Germany. 16
3. Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84, Les Diablerets, Switzerland, January 23–26, 2005. Springer, Berlin, Heidelberg, Germany. 3, 6, 12, 16, 17
4. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 629–658, Warsaw, Poland, March 23–25, 2015. Springer, Berlin, Heidelberg, Germany. 4, 6, 16, 17
5. José Becerra, Vincenzo Iovino, Dimiter Ostrev, Petra Sala, and Marjan Skrobot. Tightly-secure PAK(E). In Srdjan Capkun and Sherman S. M. Chow, editors, *CANS 17: 16th International Conference on Cryptology and Network Security*, volume 11261 of *Lecture Notes in Computer Science*, pages 27–48, Hong Kong, China, November 30 – December 2, 2017. Springer, Cham, Switzerland. 4, 17
6. Mihir Bellare. Practice-oriented provable-security. In Eiji Okamoto, George I. Davida, and Masahiro Mambo, editors, *First International Workshop on Information Security ISW '97*, volume 1396 of *LNCS*, pages 221–231. Springer, 1997. 17
7. Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 16
8. Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. 5, 18
9. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Heidelberg, Germany. 3, 6, 12, 18
10. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Berlin, Heidelberg, Germany. 3, 4, 6, 10, 11, 26
11. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. In *27th Annual ACM Symposium on Theory of Computing*, pages 57–66, Las Vegas, NV, USA, May 29 – June 1, 1995. ACM Press. 3, 6, 12
12. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Heidelberg, Germany. 23
13. Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, and Douglas Stebila. Multi-ciphersuite security of the Secure Shell (SSH) protocol. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 369–381, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. 3
14. Karthikeyan Bhargavan, Christina Brzuska, Cédric Fournet, Matthew Green, Markulf Kohlweiss, and Santiago Zanella-Béguelin. Downgrade resilience in key-exchange protocols. In *2016 IEEE Symposium on Security and Privacy*, pages 506–525, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press. 26
15. Simon Blake-Wilson and Alfred Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe, editors, *5th International Workshop on Security Protocols*, volume 1361 of *LNCS*, pages 137–158. Springer, 1997. 3, 6, 10, 11
16. Colin Boyd, Cas Cremers, Michele Feltz, Kenneth G. Paterson, Bertram Poettering, and Douglas Stebila. ASICS: Authenticated key exchange security incorporating certification systems. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 381–399, Egham, UK, September 9–13, 2013. Springer, Berlin, Heidelberg, Germany. 5, 6, 9, 17
17. Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, second edition, 2019. 1

18. Chris Brzuska, Antoine Delignat-Lavaud, Cédric Fournet, Konrad Kohbrok, and Markulf Kohlweiss. State separation for code-based game-playing proofs. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 222–249, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Cham, Switzerland. 6, 14, 15
19. Chris Brzuska and Håkon Jacobsen. A modular security analysis of EAP and IEEE 802.11. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10175 of *Lecture Notes in Computer Science*, pages 335–365, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Berlin, Heidelberg, Germany. 3, 5, 12, 18
20. Christina Brzuska. *On the foundations of key exchange*. PhD thesis, Darmstadt University of Technology, Germany, 2013. 14
21. Christina Brzuska, Marc Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. Less is more: relaxed yet composable security notions for key exchange. *International Journal of Information Security*, 12(4):267–297, 2013. 3, 6, 12
22. Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of Bellare-Rogaway key exchange protocols. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011: 18th Conference on Computer and Communications Security*, pages 51–62, Chicago, Illinois, USA, October 17–21, 2011. ACM Press. 2, 3, 4, 6, 12, 14, 15, 16
23. Christina Brzuska, Nigel P. Smart, Bogdan Warinschi, and Gaven J. Watson. An analysis of the EMV channel establishment protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 373–386, Berlin, Germany, November 4–8, 2013. ACM Press. 3, 12
24. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Heidelberg, Germany. 3, 4, 6, 9, 12
25. Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 143–161, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Heidelberg, Germany. 3, 12
26. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Berlin, Heidelberg, Germany. 2, 14
27. Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: practical issues in cryptography. In Raphael C.-W. Phan and Moti Yung, editors, *Paradigms in Cryptology - Mycrypt 2016*, volume 10311 of *LNCS*, pages 21–55. Springer, 2016. 4, 16
28. Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another look at tightness. In Ali Miri and Serge Vaudenay, editors, *SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 293–319, Toronto, Ontario, Canada, August 11–12, 2012. Springer, Berlin, Heidelberg, Germany. 4, 16, 17
29. Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 767–797, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. 4, 6, 17
30. Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On post-compromise security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016*, pages 164–178. IEEE Computer Society, 2016. 3
31. Cas Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 11: 6th ACM Symposium on Information, Computer and Communications Security*, pages 80–91, Hong Kong, China, March 22–24, 2011. ACM Press. 4, 17
32. Cas J. F. Cremers and Michele Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012: 17th European Symposium on Research in Computer Security*, volume 7459 of *Lecture Notes in Computer Science*, pages 734–751, Pisa, Italy, September 10–12, 2012. Springer, Berlin, Heidelberg, Germany. 3, 4, 6, 10, 11, 17
33. Cyprien Delpech de Saint Guilhem, Marc Fischlin, and Bogdan Warinschi. Authentication in key-exchange: Definitions, relations and composition. In *33rd IEEE Computer Security Foundations Symposium, CSF 2020,*, pages 288–303. IEEE Computer Society, 2020. 13, 25
34. Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1197–1210, Denver, CO, USA, October 12–16, 2015. ACM Press. 3, 6, 12, 16
35. Benjamin Dowling and Kenneth G. Paterson. A cryptographic analysis of the WireGuard protocol. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18: 16th International Conference on Applied Cryptography and Network Security*, volume 10892 of *Lecture Notes in Computer Science*, pages 3–21, Leuven, Belgium, July 2–4, 2018. Springer, Cham, Switzerland. 18
36. Michèle Feltz and Cas Cremers. On the limits of authenticated key exchange security with an application to bad randomness. Cryptology ePrint Archive, Report 2014/369, 2014. 5
37. Michèle Feltz and Cas Cremers. Strengthening the security of authenticated key exchange against bad randomness. *Des. Codes Cryptography*, 86(3):481–516, 2018. 4

38. Marc Fischlin and Felix Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 1193–1204, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. 3, 12, 16
39. Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In *2016 IEEE Symposium on Security and Privacy*, pages 452–469, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press. 26
40. Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271, Nara, Japan, February 26 – March 1, 2013. Springer, Berlin, Heidelberg, Germany. 11
41. Kai Gellert, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. On optimal tightness for key exchange with full forward secrecy via key confirmation. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 297–329, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. 17
42. Wesley George and Charles Rackoff. Rethinking definitions of security for session key agreement. Cryptology ePrint Archive, Report 2013/139, 2013. 3, 5, 6, 14, 18
43. Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 95–125, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. 3, 4, 16, 17
44. Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 273–293, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Heidelberg, Germany. 3, 16
45. Tibor Jager, Martijn Stam, Ryan Stanley-Oakes, and Bogdan Warinschi. Multi-key authenticated encryption with corruptions: Reductions are lossy. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 409–441, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland. 16
46. Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. One-round protocols for two-party authenticated key exchange. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 220–232, Yellow Mountain, China, June 8–11, 2004. Springer, Berlin, Heidelberg, Germany. 3, 12
47. Kazukuni Kobara, SeongHan Shin, and Mario Strefler. Partnership in key exchange protocols. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *ASIACCS 09: 4th ACM Symposium on Information, Computer and Communications Security*, pages 161–170, Sydney, Australia, March 10–12, 2009. ACM Press. 3, 6
48. Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Heidelberg, Germany. 3, 6
49. Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Heidelberg, Germany. 11
50. Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Heidelberg, Germany. 3, 12
51. Caroline Kudla and Kenneth G. Paterson. Modular security proofs for key agreement protocols. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 549–565, Chennai, India, December 4–8, 2005. Springer, Berlin, Heidelberg, Germany. 12
52. Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16, Wollongong, Australia, November 1–2, 2007. Springer, Berlin, Heidelberg, Germany. 4, 6, 9, 10
53. Kristin Lauter and Anton Mityagin. Security analysis of KEA authenticated key exchange protocol. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 378–394, New York, NY, USA, April 24–26, 2006. Springer, Berlin, Heidelberg, Germany. 3
54. Jooyoung Lee and Je Hong Park. Authenticated key exchange secure under the computational Diffie-Hellman assumption. Cryptology ePrint Archive, Report 2008/344, 2008. 17, 18, 23, 24, 25
55. Yong Li and Sven Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1343–1360, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. 3, 6
56. Alfred Menezes and Berkant Ustaoglu. Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard. In Masayuki Abe and Virgil Gligor, editors, *ASIACCS 08: 3rd ACM Symposium on Information, Computer and Communications Security*, pages 261–270, Tokyo, Japan, March 18–20, 2008. ACM Press. 3, 6

57. Jiaxin Pan, Doreen Riepel, and Runzhi Zeng. Key exchange with tight (full) forward secrecy via key confirmation. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VII*, volume 14657 of *Lecture Notes in Computer Science*, pages 59–89, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. 17
58. Phillip Rogaway. On the of role of definitions in and beyond cryptography. In *ASIAN*, volume 3321 of *LNCS*, pages 13–32. Springer, 2004. 16, 17
59. Phillip Rogaway and Till Stegers. Authentication without elision: Partially specified protocols, associated data, and cryptographic models described by code. In *22nd IEEE Computer Security Foundations Symposium, CSF 2009*, pages 26–39. IEEE Computer Society, 2009. 3, 12
60. Phillip Rogaway and Yusi Zhang. Simplifying game-based definitions - indistinguishability up to correctness and its application to stateful AE. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. 5, 9, 18
61. Victor Shoup and Aviel D. Rubin. Session key distribution using smart cards. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 321–331, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Heidelberg, Germany. 3, 12
62. Marjan Skrobot and Jean Lancrenon. On composability of game-based password authenticated key exchange. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018*, pages 443–457. IEEE, 2018. 14, 16

## A Case Study: Analyzing NAXOS+ in our model

In this section we showcase the security model we defined in Section 2 by providing a proof of the NAXOS+ protocol [54]. In fact, we consider a slight variant of NAXOS+ we call NAXOS++, whose only difference is that it includes the full protocol transcript into the key derivation function.

Specifically, in NAXOS++, each party $U$ has a long-term Diffie-Hellman key pair over a group $\mathbb{G}$ of prime order $p$, generated by a generator $g$, i.e.:

$$\mathrm{KG}() \mapsto (\mathsf{sk}, \mathsf{pk}) : a \leftarrow\!\!\$\ \{1, .., p\}, \mathsf{sk} \leftarrow a, \mathsf{pk} \leftarrow g^a$$

Each party draws an ephemeral random string $\mathsf{esk}$ and computes its ephemeral Diffie-Hellman exponent using the NAXOS trick, i.e.:

$$\begin{aligned}
&\textsc{New}(U, \mathsf{sk}_U, \mathsf{pk}_U, \mathsf{role}, V, \mathsf{PK}) \mapsto (\pi, m) : \\
&\quad \mathsf{esk} \leftarrow\!\!\$\ \{1, .., p\}, \\
&\quad \textbf{if } \mathsf{role} = \mathsf{init} : x \leftarrow H_1(\mathsf{esk}, \mathsf{sk}_U),\ X \leftarrow g^x,\ m \leftarrow X \\
&\quad \textbf{if } \mathsf{role} = \mathsf{resp} : y \leftarrow H_1(\mathsf{esk}, \mathsf{sk}_U),\ Y \leftarrow g^y,\ m \leftarrow \bot
\end{aligned}$$

The initiator and responder then both compute all possible combinations of Diffie-Hellman secrets and hash them together with their long-term public-keys and the ephemeral Diffie-Hellman public shares (including these public-shares is somewhat redundant, but it makes the key material uniqueness argument for NAXOS++ a little easier than the analogous argument about NAXOS+), i.e.:

$$\begin{aligned}
&\textsc{Run}(\pi, m) \mapsto (\pi', m') : \quad // \ \pi.\mathsf{owner} = U, \pi.\mathsf{pid} = V \\
&\quad \textbf{if } \mathsf{role} = \mathsf{init} : \\
&\qquad m' \leftarrow \bot,\ \textbf{parse } Y \leftarrow m \\
&\qquad \mathsf{keymat} \leftarrow (\mathsf{pk}_U, \mathsf{pk}_V, X, Y, \mathsf{pk}_V^{\mathsf{sk}_U}, Y^{\mathsf{sk}_U}, \mathsf{pk}_V^{x}, Y^{x}) \\
&\quad \textbf{if } \mathsf{role} = \mathsf{resp} : \\
&\qquad m' \leftarrow Y,\ \textbf{parse } X \leftarrow m \\
&\qquad \mathsf{keymat} \leftarrow (\mathsf{pk}_V, \mathsf{pk}_U, X, Y, \mathsf{pk}_V^{\mathsf{sk}_U}, \mathsf{pk}_V^{y}, X^{\mathsf{sk}_U}, X^{y}) \\
&\quad k \leftarrow H_2(\mathsf{keymat})
\end{aligned}$$

We now state our theorem for NAXOS++ $\Pi = (\mathrm{KG}, \textsc{New}, \textsc{Run})$ and observe that, although we prove security for an arbitrary number of $\textsc{Test}$ sessions, the bounds and the conceptual reduction arguments remain essentially the same as in Lee and Park [54], only the additive statistical terms Theorem 3 are slightly increased. Recall that we prove security in the tightly composable RoR model while Lee and Park [54] prove security in the 1-FtG model, cf. Figure 6.

**Theorem 3.** *For all adversaries $\mathcal{A}$, there are efficient, explicit constructions of adversaries $\mathcal{B}_1(\mathcal{A})$, $\mathcal{B}_2(\mathcal{A})$ and $\mathcal{B}_3(\mathcal{A})$ such that*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{KI}, F^{\mathsf{eCK}}}_{\Pi, n}(\mathcal{A}) \leq\ & n \cdot \mathsf{Adv}^{\mathsf{dlog}}_{\mathbb{G}, g}(\mathcal{B}_1(\mathcal{A})) \\
&+ n_s^2 \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathbb{G}, g}(\mathcal{B}_2(\mathcal{A})) \\
&+ n n_s \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathbb{G}, g}(\mathcal{B}_3(\mathcal{A})) \\
&+ \frac{q_1^2 + q_2^2 + 2n_s^3 q_2 + 2n_s^2 q_1 q_2 + n_s^2 q_2^2}{2^\lambda} \\
&+ \frac{n^2 + n_s^2 + 2n n_s q_2 + 2n n_s^2 q_1 q_2}{p}
\end{aligned} \tag{6}$$

*where $n$ denotes number of users, $n_s$ is the (total) number of sessions, $q_1$ is the number of queries to random oracle $H_1$, $q_2$ is the number of queries to random oracle $H_2$, $\mathsf{dlog}$ is the Discrete Logarithm problem, and $\mathsf{CDH}$ is the Computational Diffie-Hellman problem.*

*Proof.* The proof proceeds through a sequence of six game-hops that are summarized in Figure 7. $G_0$ is equal to $\mathbf{Exp}^{\mathsf{KI}, F^{\mathsf{eCK}}}_{\Pi, n}(\mathcal{A})$. The lines following $G_\ell$ are only executed in games $G_m$ with $m \geq \ell$. Note that in our game-hops, we use our filtering mechanism to ensure that fewer and fewer bad events can occur. Essentially, this captures "identical-up-to-bad" reasoning [12], but without the complexity of conditional probabilities. Instead, we simply modify the game such that the undesirable behavior cannot occur anymore.

From game $G_0$ to $G_1$, we remove collisions on the long-term keys. From game $G_1$ to $G_2$, we remove random oracle collisions on the random oracles $H_1$ and $H_2$. From game $G_2$ to $G_3$, we remove collisions between the

randomness that is drawn by each session. From game $G_3$ to $G_4$, we remove the case that there exists a party $U$ with long-term secret $\mathsf{sk}$ such that the adversary $\mathcal{A}$ makes a random oracle query $H_1(*, \mathsf{sk}_U)$ before, or without, making a RevLTK($U$) query. From game $G_4$ to $G_5$, we remove random oracle queries to $H_2$ with the key material of a Test session $i$ that is not partnered. From game $G_5$ to $G_6$, we remove random oracle queries to $H_2$ with the key material of a Test session $i$ that is partnered. In game $G_6$, the adversary cannot make random oracle queries to the random oracle $H_2$ with the key material of any Test session, and thus, the adversary's advantage in $G_6$ is the statistical distance between random keys and non-colliding keys (since several collisions have been removed).

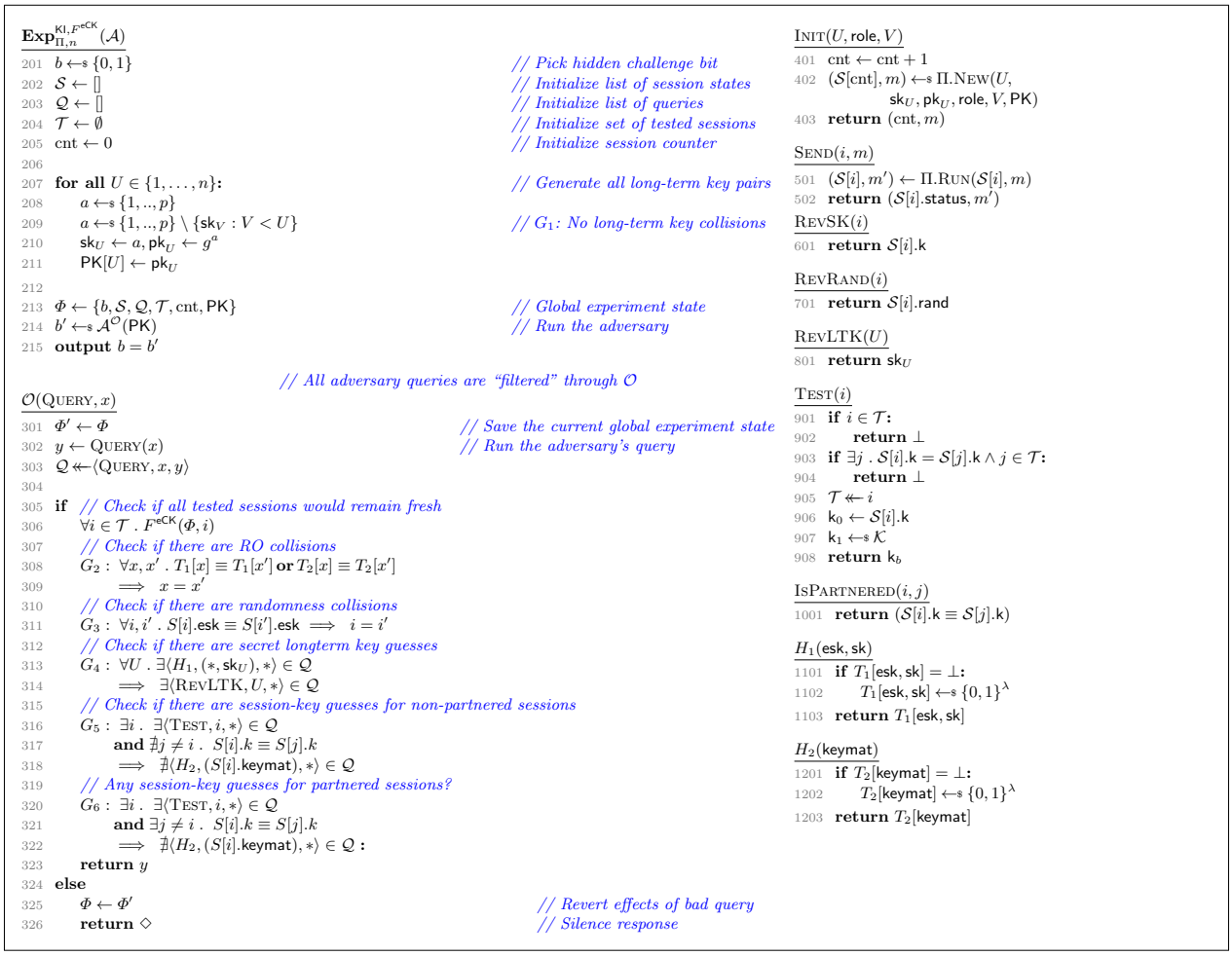


**Fig. 7:** Key Exchange Experiment for NAXOS++ protocol and freshness $F^{\mathsf{eCK}}$. The lines following $G_\ell$ are only executed in games $G_m$ with $m \geq \ell$. Recall that $\equiv$ treats two values as equal only if they have previously been defined, see Section 2.2.

We now bound the difference between each subsequent pair of games. In the following, let $\epsilon_i$ denote $\mathcal{A}$'s advantage in game $G_i$. For the first three game hops, simple collision arguments gives

$$|\epsilon_0 - \epsilon_1| \leq \frac{n^2}{p}, \; |\epsilon_1 - \epsilon_2| \leq \frac{q_1^2}{2^\lambda} + \frac{q_2^2}{2^\lambda} \text{ and } |\epsilon_2 - \epsilon_3| \leq \frac{n_s^2}{p} \,.$$

Bounding game $G_3$ and game $G_4$ is analogous to the reduction to the discrete logarithm problem (DLOG) given by Lee and Park [54]. They lose a factor $n$ when *guessing* a random party whereas we simply perform a hybrid argument over the number of parties, yielding a stronger claim with the same security loss and the same reasoning. Only the constant additive term gets increased by the hybrid argument when compared with the guessing argument.

$$\epsilon_3 - \epsilon_4 \leq n \cdot \left( \mathsf{Adv}^{\mathsf{dlog}}_{\mathbb{G},g}(\mathcal{B}_1(\mathcal{A})) + \frac{2 n_s q_2}{p} \right) \tag{7}$$

The two remaining game-hops involve reductions to the Computational Diffie-Hellman (CDH) assumption. Again, the proofs are analogous to reductions for the corresponding events in [54]. For the step from $G_4$ to $G_5$, Lee and Park [54] guess a pair of random sessions where they embed the challenge Diffie-Hellman share. We, instead of guessing them at random, perform a hybrid over all pairs. If it turns out that the pair in consideration does not end up in a a Test session, the reduction outputs a random bit (and moreover, the condition introduced for $G_4$ for this pair does not affect the game behavior, so the adversary's advantage in this hybrid step is indeed 0). For the step from $G_5$ to $G_6$, we replace guessing one party and one session by a hybrid argument over all possible combinations of a session and a party.

$$\epsilon_4 - \epsilon_5 \leq n_s^2 \cdot \left( \mathsf{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(\mathcal{B}_2(\mathcal{A})) + \frac{n_s + q_1}{2^{\lambda-1}} \right),$$

and

$$\epsilon_5 - \epsilon_6 \leq n n_s \cdot \left( \mathsf{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(\mathcal{B}_3(\mathcal{A})) + \frac{2 n_s q_1}{p} \right).$$

In game $G_6$, the adversary cannot make random oracle queries to the random oracle $H_2$ with the key material of any Test session, and thus, the adversary's advantage in $G_6$ is the statistical distance between uniformly random keys and actual keys, and $\epsilon_6$ is upper bounded by $\frac{n_s^2 q_2^2}{2^\lambda}$.

# B Authentication

To be useful, a key exchange protocol typically needs to provide authentication guarantees in addition to key-indistinguishability. For example, when a key is locally accepted by some session, it should be clear who else (if anyone) is in possession of the same key. One can also demand that one or both parties involved in the exchange are authenticated (mutual vs. one-way), that the guarantee holds as soon as the exchange has ended or when the key is actually used (explicit vs. implicit). Furthermore, the guarantees can be considered under a variety of trust assumptions where the adversary can corrupt long term keys of parties or not (i.e. key-compromise impersonation attacks) and can corrupt ephemeral keys.

In this section we show how authentication guarantees can be expressed as security predicates in our model. However, an exhaustive treatment of all different combinations is outside the scope of this paper, and we refer to de Saint Guilhem, Fischlin and Warinschi [33] for a thorough survey. We focus on providing definitions for some minimal set of authentication / agreement guarantees which we would normally expect to be satisfied.

## B.1 Implicit authentication

We start with a minimal agreement guarantee we would expect a good key-exchange protocol to satisfy: entity agreement, a.k.a. implicit authentication. Implicit authentication is a useful property to prove in addition to secrecy of the session key. We demand that if an accepted sessions has a partner, then that partner should be at the session's intended peer. Our formulation uses key-partnering.

$$\mathsf{ImplAuth}_F\colon \forall i \neq j \;.\; \Big( \big( \mathcal{S}[i].\mathsf{status} = \mathsf{accepted} \wedge \mathcal{S}[i].\mathsf{k} = \mathcal{S}[j].\mathsf{k} \wedge F(\Phi, i) \big) \implies \mathcal{S}[i].\mathsf{pid} = \mathcal{S}[j].\mathsf{owner} \Big) \quad (8)$$

**Definition 10 (Implicit authentication).** *For a freshness condition $F$ and number of parties $n \in \mathbb{N}$, a protocol $\Pi$ provides* $\epsilon$-implicit authentication *against an adversary $\mathcal{A}$ if*

$$\mathsf{Adv}^{\mathsf{ImplAuth},F}_{\Pi,n}(\mathcal{A}) := \Pr\left[ \mathbf{Exp}^{\mathsf{ImplAuth}_F,F}_{\Pi,n}(\mathcal{A}) \Rightarrow 1 \right] \leq \epsilon \;.$$

We stress that our formulation is generic in that it does not fix any particular freshness predicate. Different instantiations of the freshness lead to (substantially) different guarantees. E.g., if freshness allows the adversary to compromise the long term key of the owner of $\mathcal{S}[i]$, then the notion captures security against key-compromise impersonation (KCI) attacks. If freshness allows for the intended partner of $\mathcal{S}[i]$ to be compromised, then one captures unknown-key share attacks under this more liberal corruption model.

Furthermore, our requirement for implicit authentication is minimal. It only demands that partners agree upon the protocol participants. However, it is straightforward to extend this agreement property to cover additional variables. For instance, to ensure that the participants have different *roles* in the protocol, one can add this as an extra requirement to the ImplAuth event. In fact, agreement could be used to define a more fine-grained version of matching conversations, by demanding that partners should agree upon specific parts of

their communication transcripts (and possibly leaving other parts open to manipulation). E.g., see the use of agreement related to transcripts and downgrade attacks in [14].

Additionally, the supposition of ImplAuth depends on key partnering. It would be possible to formulate ImplAuth to depend on a generic partnering mechanism, which would potentially imply subtly different authentication properties. However, as Appendix B.3 notes, our baseline theorem of key exchange partnering implies that implicit authentication under key-partnering or a generic partnering mechanism behave similarly if the partnering mechanism is sound and inverse-sound.

Finally, we note that implicit authentication does not provide any meaningful guarantees for protocols that do not satisfy key secrecy, since in such a case no meaningful authentication is achieved.

### B.2 Explicit Authentication

The "implicit" aspect of implicit authentication means that a partner session satisfying the requirements is not actually guaranteed to exist. Some protocols also provide an explicit assurance that such a partner session exists: this is *explicit* entity authentication [10].

Explicit authentication demands that when a session $i$ accepts, it is partnered with a session of the intended partner $j$. For stateless protocols, a minimal requirement to achieve this is that the intended peer is not corrupted before session $i$ accepts. We capture this property via the predicate $F_{\mathsf{PNC}}$ below, in which the antecedent of the implication identifies the acceptance in the list of queries, and the consequent excludes any preceding long-term key reveals for the peer:

$$F_{\mathsf{PNC}}(\Phi, i)\colon \forall r < s \;.\; \big(\mathcal{Q}[s] = \langle \textsc{Send}, (i, *), (\mathsf{accepted}, *)\rangle \implies \mathcal{Q}[r] \neq \langle \textsc{RevLTK}, \mathcal{S}[i].\mathsf{peerID}, *\rangle\big). \quad (9)$$

We can then state explicit authentication as:

$$\mathsf{ExplAuth}\colon \forall i \;.\; \Big(\big(\mathcal{S}[i].\mathsf{status} = \mathsf{accepted} \wedge F_{\mathsf{PNC}}(\Phi, i)\big) \implies \exists j \neq i \;.\; \big(\mathcal{S}[i].\mathsf{k} = \mathcal{S}[j].\mathsf{k} \wedge \mathcal{S}[i].\mathsf{pid} = \mathcal{S}[j].\mathsf{owner}\big)\Big) \quad (10)$$

Notice that the predicates which define implicit authentication and explicit authentication have potentially different trust assumptions: ImplAuth allows for an arbitrary freshness predicate, while ExplAuth hard-codes the requirement that the intended peer was not corrupted before the session accepted. This is because implicit agreement guarantees can make sense even if the intended partner of session $\mathcal{S}[i]$ is corrupt (for example, with eCK-type protocols), but do not make sense for explicit authentication.

The following weaker intermediate property captures just the aliveness property of authentication: when a session accepts, and the intended partner is not corrupt, a session of the peer exists.

$$\mathsf{Alive}\colon \forall i \;.\; \Big(\big(\mathcal{S}[i].\mathsf{status} = \mathsf{accepted} \wedge F_{\mathsf{PNC}}(\Phi, i)\big) \implies \exists j \;.\; \big(\mathcal{S}[i].\mathsf{pid} = \mathcal{S}[j].\mathsf{owner}\big)\Big) \quad (11)$$

*Relationship with key-confirmation.* Another property which is sometimes mentioned alongside explicit authentication is *key-confirmation*. I.e., if a session accepts a key, then it is assured that some other session must also have computed the same key.[14] Intuitively, if key-confirmation is combined with a protocol that provides secrecy and implicit authentication, explicit authentication is achieved. Note that secrecy is strictly required here: a protocol that satisfies implicit authentication with an added key-confirmation step need not achieve explicit authentication.

### B.3 Falsifiability and partnering for authentication properties

A consequence of the baseline theorem of key exchange partnering (Theorem 1) is the following. Let $\Pi$ be a protocol, let $\phi$ be one of the three authentication properties in this section, and let $P$ be a sound and inverse-sound partnering mechanism. Then, $\Pi$ provides $\phi$ under key-partnering if and only if it provides $\phi$ under $P$-partnering.[15]

One might initially think that one can prove falsifiability of some of the authentication properties relying only on one of soundness or inverse-soundness. E.g., consider implicit authentication. In the ImplAuth predicate, the key equality-partnering check is in the supposition of the predicate. So in order to argue, e.g., that, if implicit authentication holds with $P$-partnering, it also holds with key-partnering, one might think that it suffices to have soundness: the set of sessions satisfying the supposition of the ImplAuth predicate under key partnering would

[14] Modulo some technicalities regarding which session sent/received the last message of the protocol. See [39] for a more extensive treatment of key-confirmation, including these details.

[15] Strictly speaking, to apply the baseline theorem, we need to consider the adaptations of ImplAuth, ExplAuth to $P$-partnering rather than key-partnering, by replacing the key equality checks $\mathcal{S}[i].\mathsf{k} = \mathcal{S}[j].\mathsf{k}$ on lines (8) and (10) with a partnering check $P(\mathcal{S}[i], \mathcal{S}[j])$.

then be a subset of the set of sessions satisfying the predicate under $P$-partnering. However, we cannot consider the ImplAuth predicate in isolation: while the predicate itself only uses partnering in one way, there are other parts of the overall security experiment which use partnering in various ways. In particular, the IsPartnered oracle allows the adversary to exactly learn the partner status of every session under the partnering mechanism in use (key partnering or $P$-partnering), which means we must have both soundness and inverse-soundness to guarantee the whole experiment behaves identically.

## C Changelog

Here we list the main differences between the publicly available versions.

– v12.0, July 2024: After a “slightly” protracted process, released the first public version.