A preliminary version of this paper appears in the proceedings of the *24th European Symposium on Research in Computer Security (ESORICS 2019)*, DOI: 10.1007/978-3-030-29962-0_25. This is the full version.

Breakdown Resilience of Key Exchange Protocols: NewHope, TLS 1.3, and Hybrids

Jacqueline Brendel[1] Marc Fischlin[1] Felix Günther[2]

[1] Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany
[2] Department of Computer Science and Engineering, University of California San Diego, La Jolla, USA
jacqueline.brendel@cryptoplexity.de marc.fischlin@cryptoplexity.de mail@felixguenther.info

Abstract. Broken cryptographic algorithms and hardness assumptions are a constant threat to real-world protocols. Prominent examples are hash functions for which collisions become known, or number-theoretic assumptions which are threatened by advances in quantum computing. Especially when it comes to key exchange protocols, the switch to quantum-resistant primitives has begun and aims to protect today's secrets against future developments, moving from common Diffie–Hellman-based solutions to Learning-With-Errors-based approaches, often via intermediate hybrid designs.

To this date there exists no security notion for key exchange protocols that could capture the scenario of breakdowns of arbitrary cryptographic primitives to argue security of prior or even ongoing and future sessions. In this work we extend the common Bellare–Rogaway model to capture *breakdown resilience* of key exchange protocols. Our extended model allows us to study security of a protocol even in case of unexpected failure of employed primitives, may it be number-theoretic assumptions, hash functions, signature schemes, key derivation functions, etc. We then apply our security model to analyze two real-world protocols, showing that breakdown resilience for certain primitives is achieved by both an authenticated variant of the post-quantum secure key encapsulation mechanism NewHope (Alkim et al.) which is a second round candidate in the Post Quantum Cryptography standardization process by NIST, as well as by TLS 1.3, which has recently been standardized as RFC 8446 by the Internet Engineering Task Force. Finally, we analyze the security of a generic hybrid key exchange protocol, formally showing how such designs ensure resilience against breakdowns of one of their key exchange components.

Contents

1 Introduction

Modern designs of cryptographic protocols are accompanied by a security proof which reduces the security of the protocol to the security of the employed cryptographic primitives. The security guarantees for the protocol are ultimately tied to the security of each individual primitive: with only one of the primitives being broken, all bets are usually off. However, the actual security guarantees that remain may vary with the protocol under consideration.

Key exchange protocols in particular often rely on a significant number of cryptographic primitives and hardness assumptions (e.g., collision resistant hash functions, unforgeable signature schemes, Diffie–Hellman-type assumptions, etc.). Yet, not all of them may contribute equally to the protocol's overall security at every point in time. While in general it is indeed expected that future sessions are vulnerable once the security of a component in a key exchange is broken, the question is: what can we say about the secrecy of sessions established prior to that breakdown? For special protocol designs with built-in resilience to component failures, we may even be asking for security of ongoing and future sessions if a subset of components breaks down. The notions of forward secrecy [Gün90, DVOW92, CK01] and post-compromise security [CCG16] answer these questions only partially, as we will see, for the usage of long-term secrets. A comprehensive notion of security against breakdowns of arbitrary (keyed and unkeyed) primitives as well as cryptographic hardness assumptions is however lacking.

1.1 Breakdowns and Mitigations in Real-World Key Exchange

The absence of a precise understanding of primitive breakdowns is despite such disruptions being an ever present threat, through failures or significant weakening of cryptographic algorithms and assumptions. With computational and cryptanalytic capabilities steadily evolving, examples of such incidences abound and range from weak ciphers like RC4 [GMPS14, ABP+13] over poor Diffie–Hellman parameter choices [ABD+15] to advances in breaking widely deployed hash functions like MD5 [dB94, WY05, SLdW07] or SHA-1 [WYY05, Ste13, SKP16, SBK+17] enabling key-exchange-level attacks [BL16].

Moreover, the anticipated advent of quantum computers promises to render many of the currently used cryptographic algorithms and hardness assumptions obsolete. To remedy this situation, post-quantum secure schemes and in particular key exchange protocols are already developed (e.g., [BCNS15, ADPS16b, BCD+16, BDK+18]) and have in parts been experimentally deployed (e.g., [Bra16, Lan18]). However, often only the most crucial cryptographic algorithms are replaced by post-quantum secure alternatives. Other components of the protocol, especially signature schemes, remain "classical" for the time being. The reasoning behind this is that exploits for these components would need to happen during the protocol execution to enable attacks on the key exchange, and not only once quantum computers reach maturity. Indeed, for example, the authors of the quantum-secure key-exchange protocol NewHope argue that "[...] attacks on the [classical] signature will not compromise previous communication" [ADPS16b]. While this intuition may be correct, there are no formal justifications for such statements at this point.

Until full confidence in the recently proposed post-quantum schemes and their parameter selection is established (see, e.g., the NIST post-quantum cryptography standardization effort [NIS17]), so-called *hybrid* schemes are seen as a suitable way to guard today's communications from "record-today-then-break-later" adversaries, which are often referred to as future quantum adversaries. These key exchange schemes combine classical and post-quantum secure mechanisms such that the resulting session key remains secure as long as one of the two (or more) components remains secure. Academia has recently investigated how to build such hybrid schemes, e.g., from KEM combiners [GHP18, BBF+19]. While [GHP18] solely treats the construction of KEM combiners, [BBF+19] additionally introduces security notions for hybrid authenticated key exchange with respect to quantum adversaries, but still focuses exclusively on KEM-based protocols.

1.2 Our Contributions

There is hence a need for a generic formal tool to assess the precise security of key exchange protocols in case (some) arbitrary underlying primitives or hardness assumptions break.

In this work, we introduce a novel security model that captures Bellare–Rogaway-style key exchange security under the breakdown of cryptographic primitives and assumptions. We then study the post-quantum design NewHope by Alkim et al. [AAB+19] as submitted to the NIST post-quantum standardization process as well as the Transport Layer Security (TLS) protocol in its latest version 1.3 [Res18] with respect to their resilience (of past communication) against such breakdowns. Our analyses formally confirm some of the intuition above, but also exhibit that seemingly minor technical design choices can unforeseenly impair the breakdown resilience of protocols.

Through a stronger version of our model, we furthermore capture the impact of cryptographic breakdowns on ongoing and future sessions. While classical key exchange protocols in general guarantee no security in this setting, for special designs such as hybrid protocols, this notion becomes meaningful. We demonstrate this by showing how hybrid key exchange designs can satisfy this stronger notion of breakdown resilience.

Security model for breakdown resilience. To provide a formal ground for analyses concerning the effects primitive breakdowns have on key exchange protocols, we propose a formal key exchange security model in Section 3 capturing such breakdowns as an extension to the well-established model by Bellare and Rogaway [BR94] (which we first recap in Section 2).

In our model, we concretely formalize the effects resulting from the breakdown of some security properties of (instances of) cryptographic primitives or hardness assumptions by specifying the additional capabilities the adversary gains through such a breakdown.[1] For example, we model the breakdown of an encryption scheme by granting the adversary access to secret keys, or the breakdown of collision resistance of a hash function by enabling the adversary (from the point of breakdown onwards) to define inputs to the hash function to collide arbitrarily. Our model can generically handle other choices for consequences of breakdowns. The conservative choice here of considering strong break capabilities, such as being able to find arbitrary collisions, makes the adversary more powerful and thus provides stronger security guarantees of resistant protocols. It also saves us from specifying dedicated break possibilities for different protocols, potentially proving breakdown resistance in one case, while being susceptible to attacks in other protocols. We however stress that our model flexibly supports tailored choices for break capabilities, e.g., if one prefers to consider a hash function's security degradation in a more fine-grained manner by distinguishing, say, arbitrary and structured collisions.

The resulting security notion of *breakdown resilience* (for a specified set of primitives and corresponding security assumptions) then demands that keys established in sessions prior to the point of breakdown remain secure. That is, such keys should still be indistinguishable from random for the adversary, even when capable of breaking the given primitives.[2] It turns out that "half-open" sessions need a special treatment in this regard, in order to also appropriately capture active attacks on past sessions within the model. Later (in Section 6), we furthermore consider a *strong* breakdown resilience variant (for specifically designed protocols such as hybrid constructions), which demands security even for ongoing and future sessions.

We formalize breakdowns via an additional Break oracle provided to the adversary beyond the classical

[1] For any protocol in practice, it will be a specific *instance* (e.g., MD5 or SHA-1) of a class of primitives (e.g., hash functions) that is weakened and whose security property breaks down. Our model accordingly allows to distinguish breakdowns of, e.g., the primitive (instance) MD5 and the primitive (instance) SHA-1.

[2] Naturally, we consider any breakdown of a cryptographic component devastating for the employing key exchange protocol's future security (as the component could be omitted otherwise), thus we demand security only for previously completed sessions.

oracles given in a Bellare–Rogaway-style key exchange model. When invoked, the Break oracle fixes the point in time of the breakdown and grants the adversary with a response and/or further oracle accesses, enabling it to break the set of primitives and hardness assumptions specified as a parameter of the model. As we will see, this mechanism is extremely versatile. Along with our model, we provide possible descriptions for the behavior of Break for a number of cryptographic primitives and assumptions commonly employed in key exchange protocols, including encryption and signature schemes, hash functions, key derivation functions, hardness of the discrete logarithm problem, and more. Most importantly, however, our Break oracle can easily be extended to capture further primitives or different types of security-assumption breakdowns by simply specifying the information provided to the adversary in case of a breakdown of that primitive or assumption.

Breakdown resilience of NewHope. We then exercise our model in Section 4 on an authenticated variant of NewHope-Nist, a post-quantum key encapsulation mechanism proposed by Alkim et al. [AAB+19] and submitted to the NIST Post Quantum Cryptography standardization effort. We first define Auth-NewHope as a natural, authenticated version of the IND-CPA secure NewHope-Nist KEM from [AAB+19] by employing authentication via (classical) signatures and MACs following the SIGMA (SIGn-and-MAc) approach proposed by Krawczyk [Kra03], which has been adopted in major Internet security protocols like IPsec and TLS.

Using our new formalism, we confirm the intuition that, in particular, a signature breakdown does not compromise the security of prior completed sessions. For this, we provide a security proof in our model, establishing breakdown resilience for both signature and MAC unforgeability. As the Auth-NewHope protocol employs a generic SIGMA-style [Kra03] authentication step following the basic NewHope-Nist key encapsulation, our results can furthermore be seen as a validation of the breakdown resilience (for signatures and MACs) achieved by applying SIGMA-style authentication to an unauthenticated key establishment protocol as a compiler.

Breakdown resilience of TLS 1.3. As the second example, we assess in Section 5 the breakdown resilience of the key exchange (the so-called handshake) of TLS 1.3, the latest version of the Transport Layer Security protocol recently standardized as RFC 8446 [Res18]. To this end, we consider two major handshake modes, the full (elliptic-curve) ephemeral-Diffie–Hellman ((EC)DHE) handshake as well as the resumption-style (PSK) handshake based on pre-shared keys.

For the (EC)DHE handshake, we prove breakdown resilience for collision resistance of the hash function used to compute transcript hashes (for key derivation, signatures, etc.) as well as unforgeability of both the employed signature and MAC scheme. In our analysis, we restrict ourselves to the security of the main application data key established in a mutually authenticated handshake, omitting more advanced features of TLS 1.3 in order to focus our attention on the achieved breakdown resilience properties.

For the PSK(-only) handshake, we determine that—perhaps surprisingly at first glance—no breakdown resilience at all is provided. This is despite the PSK mode following a similar structure as the full handshake and hence possibly raising hope for similar resistance to a hash function breakdown (signatures are not used in the PSK mode and MACs do not contribute to its security). However, for reasons rooted in technical details of the key derivation schedule which we will discuss, hash collision attacks can lead to a complete break of the PSK mode's key exchange security. Along with this negative result, we discuss both mitigations and practical concerns, as well as why including ephemeral Diffie–Hellman shares (in the combined PSK-(EC)DHE handshake mode) is favorable for not only providing forward secrecy but also recovering breakdown resilience (for the employed hash function and MAC scheme).

Strong breakdown resilience of hybrid constructions. We finally introduce an even stronger variant of breakdown resilience that demands security of session keys under component breakdowns even in ongoing and future sessions. This specifically enables us to argue about the security of hybrid key exchange designs which we illustrate through analyzing generic hybrid constructions under one-out-of-two component breakdowns.

1.3 Related Work and Delineation

Our work extends, and is inspired by, conceptual ideas of prior work on the security of both key exchange specifically and cryptographic protocols more broadly. Yet, our notion of breakdown resilience is novel and unmet by any (combination of) previously defined security goals, as we discuss in the following.

Forward secrecy. While similar in spirit, breakdown resilience should not be confused with the concept of forward secrecy [Gün90, DVOW92, CK01]. Forward secrecy as a security property of session keys derived in a key exchange protocol demands that even if an involved party's long-term secret is compromised, any key derived previously remains secure. While this property is closely related to our scenario, breakdown resilience takes a conceptually distinct approach to forward secrecy (and also stronger security models allowing ephemeral key reveal [CK01, LLM07]): its focus is on the breakdown of complete primitives or hardness assumptions rather than on the exposure of specific protocol values like long-term keys. Furthermore, breakdown resilience also covers breaks of unkeyed cryptographic building blocks (e.g., breaking collision resistance of hash functions) and more generally cryptographic hardness assumptions such as the discrete logarithm problem.

To make the distinction even more explicit, consider a KEM-based key exchange protocol [BCGP08] like the scheme based on the Kyber KEM [BDK+18]. In such schemes a static KEM instance usually serves authentication purposes and an ephemeral KEM instance, based on the same hardness assumption, is used to establish the key and to provide forward secrecy. A breakdown of the underlying KEM assumption, however, would also reveal the secret keys of all past sessions. This demonstrates that not all effects of future compromises of keyed primitives can be captured through the notion of forward secrecy, let alone breakdowns of unkeyed primitives or assumptions.

Post-compromise security. With their notion of *post-compromise security*, Cohn-Gordon, Cremers, and Garratt [CCG16] establish security guarantees for communication *after* participants have been compromisedto various degrees. (Strong) breakdown resilience differs from this notion in that it considers not the compromise of single parties but the global breakdown of cryptographic building blocks on a protocol level. Strong breakdown resilience may be seen as a generalization of the concept of post-compromise security while our standard notion is concerned with the security of sessions that were completed *before* a breakdown occurred.

Bitcoin security in the presence of broken primitives. Giechaskiel, Cremers, and Rasmussen [GCR16] were the first to systematically explore how broken or weakened hash functions and/or signatures affect the security of Bitcoin. While their study focused on Bitcoin, we present a general framework that can be applied to analyze a whole class of cryptographic protocols, namely authenticated key exchange protocols, and may very well be transferable to other kinds of protocols.

Downgrade resilience. A breakdown of a primitive or hardness assumption willingly employed by both parties conducting a key exchange is conceptually different from a downgrade of a connection to an insecure cipher suite during the negotiation phase. In the breakdown resilience setting we are concerned with the

security of past sessions after a breakdown has occurred, while downgrade resilience, formally treated by Bhargavan et al. [BBF+16] and Dowling and Stebila [DS15], assures that weak cipher suites will never be successfully negotiated in case matching stronger suites are preferred by both participants.

Security analyses of NewHope and TLS 1.3. Prior work on the security of NewHope focused on the security as an unauthenticated key exchange protocol [ADPS16b]. We augment NewHope to include authentication and study its security not only as an AKE protocol but also with respect to breakdown resilience.

TLS 1.3 has received substantial attention from the research community on its way to standardization; we specifically point to analyses of the handshake protocol in both computational or symbolic models as well as through formal verification [DFGS15, DFGS16, CHSvdM16, FGSW16, LXZ+16, Kra16, FG17, BBK17, CHH+17] and also refer to [PvdM16] for a review of the standardization process. In this work we do not aim at providing a full key exchange security analysis of the TLS 1.3 handshake modes specified, but focus on the novel property of breakdown resilience in two main modes, (EC)DHE and PSK, which has not been studied for TLS 1.3 so far.

Hybrid key exchange. The model for hybrid authenticated key exchange proposed by Bindel et al. [BBF+19] is a Bellare–Rogaway-style model adjusted to two-stage adversaries with different levels of quantumness. Their constructions focus on hybrid key encapsulation mechanisms (KEMs), where the breakdowns are caused exclusively by these quantum adversaries. Our model for (strong) breakdown resilience offers a more general, alternative approach. It is able to explicitly capture the breakdown of multiple arbitrary primitives or even hardness assumptions, irrespective of the cause, thus in particular avoiding the complexities of a two-stage adversary setting.

2 The Bellare–Rogaway Modelfor Authenticated Key Exchange

We begin by recapping key exchange security in the style of the model by Bellare and Rogaway [BR94] which forms the basis for our model of breakdown resilience. This model provides strong security guarantees for authenticated key exchange in the presence of an active adversary. As formalized in the following, the adversary interacts with protocol instances via oracle queries with the goal to distinguish the real session key established in a 'test' session of its choice from a randomly chosen one (via a Test oracle). The adversary is considered to have full control over the network (modeled via a Send oracle delivering messages to key exchange sessions). It is furthermore able to corrupt some of the parties' long-term secrets (via a Corrupt oracle) and to reveal some of the established session keys in honest sessions (via a Reveal oracle).

In this work we focus on the case of mutually authenticated key exchange protocols with pre-specified peer identities, but note that the model can be extended to capture unilaterally authenticated or anonymous key exchange as well as post-specified peers. We furthermore distinguish between protocols providing and not providing forward secrecy.

Notation and overview. The participants in a key exchange protocol KE are given by elements U from the set of users $\mathcal{U}$, each of whom holds a long-term public key pk_U with corresponding secret key sk_U. Each participant can act as initiator or responder of a protocol execution and may run multiple instances, so-called *sessions*, of the key exchange protocol in parallel. To uniquely refer to the k-th session owned by user $U \in \mathcal{U}$ with intended communication partner $V \in \mathcal{U}$ on an administrative level, we use the notation $\pi^k_{U,V}$. Each such session is associated with the following set of variables:

- $\mathsf{role} \in \{\mathsf{initiator}, \mathsf{responder}\}$ is the session owner's role in this session.
- $\mathsf{st}_{\mathsf{exec}} \in \{\mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$ denotes the current state of execution (default upon creation: $\mathsf{running}$).
- $\mathsf{sid} \in \{0,1\}^* \cup \{\perp\}$ indicates the session identifier (default: $\perp$).
- $\mathsf{st}_{\mathsf{key}} \in \{\mathsf{fresh}, \mathsf{revealed}\}$ indicates the state of the session key K (default: fresh).
- $\mathsf{K} \in \{0,1\}^* \cup \{\perp\}$ indicates the established session key (default: $\perp$).
- $\mathsf{tested} \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether the session key K has been tested or not (default: false).

To be able to refer to a specific entry for a session $\pi^k_{U,V}$, we use the notation $\pi^k_{U,V}.\mathsf{entry}$. For example, $\pi^k_{U,V}.\mathsf{role}$ specifies the session owner U's role in session $\pi^k_{U,V}$. For simplicity, we sometimes simply write π and π' to refer to sessions in a general context where the specific indices do not matter.

Partnering of sessions. The partnering of sessions is defined via the session identifiers. More precisely, we call the session $\pi^k_{U,V}$ owned by U *partnered* with the session $\pi^{k'}_{V',U'}$ owned by V' (and vice versa), if the sessions share the same session identifier, i.e., $\pi^k_{U,V}.\mathsf{sid} = \pi^{k'}_{V',U'}.\mathsf{sid} \neq \perp$. We require that any execution between honest instances is partnered.

2.1 Adversary Model

We model the adversary as a probabilistic polynomial time (PPT) Turing machine denoted by $\mathcal{A}$. The adversary is active and in full control over the network. This implies in particular that—additional to the interception of messages—the adversary can schedule when (and if) message delivery occurs. Furthermore, the adversary may alter and inject messages. We assume the adversary learns if a participant in the protocol has terminated and/or accepted.

Adversarial queries. In order to break key secrecy, the goal of the adversary is to distinguish real from random session keys. Not all interactions of the adversary with the protocol are admissible at any point. In particular, there are conditions under which the adversary trivially loses the game, e.g., when both revealing and testing session keys of partnered sessions as mentioned before. To keep track if one of these cases has occurred, we leverage a flag lost initialized to false.

The adversary interacts with the protocol via the following oracle queries:

$\mathsf{NewSession}(U, V, role)$**:** Establishes a new session $\pi^k_{U,V}$ for U (with k being the next counter value for sessions of U with intended partner V), stores the given role value in $\pi^k_{U,V}.\mathsf{role} \leftarrow role$, and returns the identifier $\pi^k_{U,V}$.

$\mathsf{Send}(\pi^k_{U,V}, m)$**:** Causes the message m to be sent to the session $\pi^k_{U,V}$. If there exists no session $\pi^k_{U,V}$, the query outputs $\perp$. Else the response of the session owner U upon receipt of message m is returned, and the state of execution $\mathsf{st}_{\mathsf{exec}}$ is updated. If $\mathsf{st}_{\mathsf{exec}}$ changes to $\mathsf{accepted}$ with an intended communication partner V that was previously corrupted, then set $\mathsf{st}_{\mathsf{key}} \leftarrow \mathsf{revealed}$.

$\mathsf{Reveal}(\pi^k_{U,V})$**:** Returns the session key K of session $\pi^k_{U,V}$. If there exists no session $\pi^k_{U,V}$ or if $\mathsf{st}_{\mathsf{exec}} \neq \mathsf{accepted}$, then return $\perp$. Otherwise, set $\mathsf{st}_{\mathsf{key}}$ to $\mathsf{revealed}$ and return K to the adversary.

$\mathsf{Corrupt}(U)$**:** Returns the long-term secret key sk_U of U to the adversary. No further queries may be issued to sessions owned by U. In case of no forward secrecy, $\mathsf{st}_{\mathsf{key}}$ is set to $\mathsf{revealed}$ in all sessions $\pi^k_{V,W}$ where $V = U$ or $W = U$.

$\mathsf{Test}(\pi^k_{U,V})$**:** Tests the session key of session $\pi^k_{U,V}$. The oracle uses a test bit b_{test} chosen uniformly at random at the outset and then fixed during the game execution. For simplicity, we restrict the adversary to ask a single Test query only. If there exists no session $\pi^k_{U,V}$ or if $\pi^k_{U,V}.\mathsf{st}_{\mathsf{exec}} \neq \mathsf{accepted}$, the query returns $\perp$. Otherwise, $\pi^k_{U,V}.\mathsf{tested}$ is set to true. If $b_{\mathsf{test}} = 0$, a key $\mathcal{K} \xleftarrow{\$} \mathcal{D}$ is sampled at random from the session key distribution $\mathcal{D}$. If $b_{\mathsf{test}} = 1$, $\mathcal{K}$ in contrast is set to the actual session key $\pi^k_{U,V}.\mathsf{K}$. Return $\mathcal{K}$.

2.2 Bellare–Rogaway AKE Security Games

We adopt the approach of Brzuska et al. [BFWW11, Brz13] to separate the overall BR security properties into the notions of BR-Match security and BR key secrecy. The conditions of BR-Match security guarantee that the session identifiers sid ensure an appropriate identification of partnered sessions, that at most two sessions are partnered, and that partnered sessions hold the same key. BR key secrecy then ensures that a protocol establishes session keys that are indistinguishable from random strings and (implicitly) mutually authenticated. This, of course, excludes some trivial attacks like distinguishing revealed session keys from random keys.

Definition 2.1 (BR-Match Security)**.** *Let λ be the security parameter, KE a key exchange protocol, and $\mathcal{A}$ a PPT adversary interacting with KE via the queries defined in Section 2.1 in the following game $G^{\mathsf{BR\text{-}Match}}_{\mathsf{KE},\mathcal{A}}(\lambda)$:*

Setup. *The challenger generates long-term public/private-key pairs with certificates for each participant $U \in \mathcal{U}$.*

Query. *The adversary $\mathcal{A}$ receives the generated public keys and has access to the queries* NewSession*,* Send*,* Reveal*,* Corrupt*, and* Test*.*

Stop. *At some point, the adversary stops with no output.*

We say that $\mathcal{A}$ wins the game, denoted by $G^{\mathsf{BR\text{-}Match}}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1$, if at least one of the following conditions holds:

1. *There exist two distinct sessions π and π' with $\pi.\mathsf{sid} = \pi'.\mathsf{sid} \neq \perp$, and $\pi.\mathsf{st}_{\mathsf{exec}}, \pi'.\mathsf{st}_{\mathsf{exec}} \neq \mathsf{rejected}$, but $\pi.\mathsf{K} \neq \pi'.\mathsf{K}$. (Different session keys in partnered sessions.)*
2. *There exist two sessions $\pi := \pi^k_{U,V}$ and $\pi' := \pi^{k'}_{V',U'}$ such that $\pi.\mathsf{sid} = \pi'.\mathsf{sid} \neq \perp$, $\pi.\mathsf{role} = \mathsf{initiator}$, and $\pi'.\mathsf{role} = \mathsf{responder}$, but $U \neq U'$ or $V \neq V'$. (Different intended partner.)*
3. *There exist at least three sessions π, π', and π'' such that π, π', π'' are pairwise distinct, but $\pi.\mathsf{sid} = \pi'.\mathsf{sid}' = \pi''.\mathsf{sid} \neq \perp$. (More than two sessions share the same session identifier.)*

We say KE *is* BR-Match-secure *if for all PPT adversaries $\mathcal{A}$ the advantage function*

$$\mathsf{Adv}^{\mathsf{BR\text{-}Match}}_{\mathsf{KE},\mathcal{A}} := \Pr\left[G^{\mathsf{BR\text{-}Match}}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1\right]$$

is negligible in the security parameter λ.

Definition 2.2 (BR Key Secrecy)**.** *Let λ be the security parameter,* KE *a key exchange protocol with key distribution $\mathcal{D}$, and $\mathcal{A}$ a PPT adversary interacting with* KE *via the queries defined in Section 2.1 in the following game $G^{\mathsf{BR},\mathcal{D}}_{\mathsf{KE},\mathcal{A}}(\lambda)$:*

Setup. *The challenger generates long-term public/private-key pairs for each participant $U \in \mathcal{U}$, chooses the test bit $b_{\mathsf{test}} \xleftarrow{\$} \{0,1\}$ at random, and sets* $\mathsf{lost} \leftarrow \mathsf{false}$.

Query. *The adversary $\mathcal{A}$ receives the generated public keys and has access to the queries* NewSession, Send, Reveal, Corrupt, *and* Test.

Guess. *At some point, $\mathcal{A}$ stops and outputs a guess b_{guess}.*

Finalize. *The challenger sets the 'lost' flag to* $\mathsf{lost} \leftarrow \mathsf{true}$ *if there exist two (not necessarily distinct) sessions π, π' such that $\pi.\mathsf{sid} = \pi'.\mathsf{sid}$, $\pi.\mathsf{st}_{\mathsf{key}} = \mathsf{revealed}$, and $\pi'.\mathsf{tested} = \mathsf{true}$. (Adversary has tested and revealed the key in a single session or in two partnered sessions.)*

We say that $\mathcal{A}$ wins the game, denoted by $G^{\mathsf{BR},\mathcal{D}}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1$, if $b_{\mathsf{guess}} = b_{\mathsf{test}}$ and $\mathsf{lost} = \mathsf{false}$. *Note that the winning conditions are independent of the forward secrecy property of* KE*, as forward secrecy is already taken into account in the* Corrupt *query.*

We say that KE *provides* BR key secrecy *with/without forward secrecy if for all PPT adversaries $\mathcal{A}$ the advantage function*

$$\mathsf{Adv}^{\mathsf{BR},\mathcal{D}}_{\mathsf{KE},\mathcal{A}}(\lambda) := \Pr\left[G^{\mathsf{BR},\mathcal{D}}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}$$

is negligible in the security parameter λ.

Definition 2.3 (BR Security)**.** *We say a key exchange protocol* KE *is* BR-secure *(with/without forward secrecy) if* KE *provides* BR-Match *security and* BR *key secrecy (with/without forward secrecy), according to Definitions 2.1 and 2.2.*

3 Modeling Breakdown Resilience

For integrating *breakdown resilience* into the generic (Bellare–Rogaway-style) security model for authenticated key exchange, we are interested in the security of completed sessions in the case that one or multiple cryptographic primitives or hardness assumptions underlying the key exchange protocol's security break. Note that for classical key exchange designs one cannot expect any security guarantees to remain for ongoing and future sessions, as they may crucially rely on the broken primitive's security. In Section 6, we will discuss the specific class of hybrid designs which achieve a *strong* variant of breakdown resilience we define there, capturing security also of ongoing and future sessions.

Figure 1 illustrates how different scenarios are treated in our model. For now, we are interested in the question of whether the expected security level is still achieved in *past* sessions (Scenarios 1 to 3 in Figure 1) and thus exclude sessions that are still active at the time of breakdown or start after it (Scenarios 4 and 5). It is however not only the status of the test session which is crucial for the security guarantees, but also that of a potential (unfinished) communication partner, which we refer to as the associated session. A breakdown of a primitive in the middle of the communication may enable the adversary to interfere with the correct partnering of sessions, leading to trivial attacks on the session key in question, which we need to capture in our model. Consider, for example, a test session that has accepted and has output its last message, say, to authenticate itself, waiting to be delivered to its intended partner session. Such final-message authentication is indeed very common in key exchange protocols. An adversary with breakdown capabilities can now modify this last message, e.g., by forging a new signature, to cause the intended

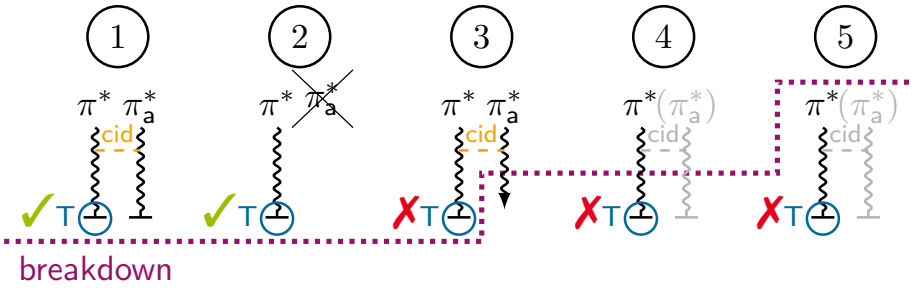


Figure 1: Illustration of (non-)permissible Test queries wrt. a breakdown. The dotted purple line indicates the point in time of a breakdown with respect to the five scenarios of (completed, running, or future) test sessions. T denotes a test query on session π^*, π^*_{a} denotes a (potential, if gray) associated session (semi-)partnered with π^* holding the same contributive identifier (cid). A checkmark ✓ (resp. a cross ✗) indicates whether the test query is admissible or not.

partner to accept with a different session identifier. Yet, the intended partner may still derive the same key as our test session as the relevant key material is already established. The adversary could hence safely learn the session key through a Reveal query on the now unpartnered session, trivially distinguishing the tested key from random. This situation is depicted in Scenario 3 of Figure 1.

Hence, we need to exclude sessions from being tested that accepted prior to the breakdown but have a "semi-completed" partner session that, at the time, already holds all the relevant cryptographic material for the final key derivation (Scenario 3). We use a notion of contributive identifiers (cid) to identify such almost-partnered sessions. Identical contributive identifiers indicate that sessions may eventually derive the same key, despite not being partnered yet.

An alternative to using contributive identifiers would be to demand that only sessions that fully completed before breakdown with an honest partner would be considered valid test sessions (as in Scenario 1). This, however, would limit the adversary to purely passive attacks in the pre-breakdown phase. In contrast, our approach with contributive identifiers is less restrictive, as we still allow the adversary to test completed sessions without an honest partner (Scenario 2), e.g., where the adversary communicated with that party.

To capture resilience against breakdowns, we augment our model with a Break query that allows the adversary to break the security of cryptographic primitives or hardness assumptions contained in a dedicated, specified set $\mathcal{F}_{\mathsf{BDR}}$. More precisely, this set has the form $\mathcal{F}_{\mathsf{BDR}} = \{(f_1, \mathsf{sec\text{-}prop}_1), (f_2, \mathsf{sec\text{-}prop}_2), \ldots\}$, i.e., $\mathcal{F}_{\mathsf{BDR}}$ contains tuples $(f, \mathsf{sec\text{-}prop})$, determining all primitives/hardness assumptions f for which some security property sec-prop may break. As a result of the Break query, the adversary may—depending on the broken security property of the primitive or assumption—be given certain key material or access to additional oracles in the model. To capture that we expect only sessions to remain secure that completed before the breakdown occurred, we introduce a flag breakdown which is set when Break is called and checked within the (accordingly modified) Send query. These changes enable us to formalize a model for breakdown resilience in a generic way. As we will see, our notion of a Break query is versatile and can capture a wide variety of breakdowns. Primitives and assumptions for which we provide a concrete specification of a breakdown include, e.g., the unforgeability of signatures, CCA security of encryption schemes, collision resistance of hash functions, or the discrete logarithm problem. Formal definitions for a number of these security properties can be found in Appendix A. As it turns out, breakdown resilience (with/without forward secrecy) provides strictly stronger security than the notion of BR security given in the previous section.

3.1 Extensions to the Security Model

In the following, we specify the formal extensions made to the basic Bellare–Rogaway-style security model from Section 2 to capture breakdown resilience.

Breakdown flag. We introduce a global flag breakdown (initialized to false) in the security game, indicating whether the adversary has issued a Break query.

Contributive identifiers. We augment the model with the concept of contributive (session) identifiers.[3] Intuitively, contributive identifiers relate two sessions which exchanged the messages establishing the key material (e.g., values g^x and g^y in a Diffie–Hellman-style protocol), but are not yet partnered (e.g., because the authenticating signatures have not been sent yet). In the breakdown setting, contributive identifiers enable us to specify that we do not expect security of sessions that, at time of breakdown, had a "semi-partnered" session that shares the same key material. The reason is that the adversary could eventually make this "semi-partnered" party accept after the breakdown for the same session key but a different session identifier, e.g., by forging the final protocol signature after the breakdown; in this case achieving key indistinguishability would be impossible. We thus demand that the tested session accepted prior to the breakdown and does not share a contributive identifier with another session that was still running at the time of breakdown.

Formally, we add the following variables associated with each session $\pi^k_{U,V}$:

- $\mathsf{cid} \in \{0,1\}^* \cup \{\bot\}$ indicates the contributive identifier (default: $\bot$).
- $\mathsf{st}^{\mathsf{bd}}_{\mathsf{exec}} \in \{\mathsf{running}, \mathsf{accepted}, \mathsf{rejected}, \bot\}$ denotes the state of execution at the time of breakdown,i.e., when the Break query was issued the first time (default prior to breakdown: $\bot$).

To avoid trivial choices and to relate the contributive identifiers (cid) to session identifiers (sid) we add two requirements for Match security: First, as in [DFGS15], same session identifiers must imply same contributive identifiers, capturing the intuition that partnered session should in particular be contributively partnered. Second, since we restrict the Test query based on common contributive identifiers, we demand that at most two sessions share the same cid to prevent that Test queries are excluded by trivial choices of colliding contributive identifiers.

Break query. We add a Break query to the adversarial queries described in Section 2.1 which allows the adversary to schedule the timing of breakdowns. The query sets breakdown to true, records the current execution state of sessions, and provides the adversary with the capability to break the security of any $(f, \mathsf{sec\text{-}prop}) \in \mathcal{F}_{\mathsf{BDR}}$, where $\mathcal{F}_{\mathsf{BDR}}$ is a fixed parameter of the security game.

Which capability the adversary is given when breaking the security sec-prop of a primitive or assumption f depends on the latter's type and may, e.g., be exposing all key material used within f to the adversary or granting it access to additional oracles. We discuss options for the common primitives below in Section 3.2 and specify the corresponding behavior of Break in Table 1. As we will see, additional primitives and assumptions can easily be added to capture further key exchange designs as the Break query itself is generic.

[3] We here use the formalization by Dowling et al. [DFGS15] from their analysis of TLS 1.3 candidate handshakes in the multi-stage key exchange setting. Contributive identifiers are furthermore related to the concept of "origin-sessions" for partnering based on matching conversation introduced by Cremers and Feltz [CF12] and the notion of (peer-)exchange variables used by Bhargavan et al. [BFK+14].

$\mathsf{Break}()$**:** Causes for all $(f, \mathsf{sec\text{-}prop}) \in \mathcal{F}_{\mathsf{BDR}}$ the breakdown of the security property $\mathsf{sec\text{-}prop}$ of the cryptographic primitive or hardness assumption f.

If $\mathsf{breakdown} = \mathsf{false}$, for all sessions π record the current state of execution as $\pi.\mathsf{st}^{\mathsf{bd}}_{\mathsf{exec}} \leftarrow \pi.\mathsf{st}_{\mathsf{exec}}$. Set $\mathsf{breakdown} \leftarrow \mathsf{true}$. Depending on the entries in the set $\mathcal{F}_{\mathsf{BDR}}$, provide the adversary with the responses and/or oracle accesses specified in Table 1. The Break oracle may be queried repeatedly, which enables the adversary to obtain an updated response in order to, e.g., receive further key material used in an encryption scheme since the last call of Break.

Modified Send query. Once the $\mathsf{breakdown}$ flag is set to true, ongoing sessions and sessions that are initiated after the breakdown must be considered revealed as we expect their keys to be affected by the breakdown. To enforce this, we replace the Send query from Section 2.1 by the following slightly modified version that sets the session key state to $\mathsf{revealed}$ if $\mathsf{breakdown} = \mathsf{true}$; the change is underlined in the following description.

$\mathsf{Send}_{\mathsf{BDR}}(\pi^k_{U,V}, m)$**:** Causes the message m to be sent to the session $\pi^k_{U,V}$. If there exists no session $\pi^k_{U,V}$, the query outputs $\perp$. Else the response of the session owner U upon receipt of message m is returned, and the state of execution $\mathsf{st}_{\mathsf{exec}}$ is updated. If $\mathsf{st}_{\mathsf{exec}}$ changes to $\mathsf{accepted}$ with an intended communication partner V that was previously corrupted <u>or if $\mathsf{breakdown} = \mathsf{true}$</u>, then set $\mathsf{st}_{\mathsf{key}} \leftarrow \mathsf{revealed}$.

3.2 Breakdown of Primitives and Assumptions

We next specify the behavior of the Break query and capabilities the adversary is provided with for a number of common cryptographic primitives and hardness assumptions. Table 1 covers a wide range of standard primitives and assumptions underlying the security of most key exchange protocols (and in particular the NewHope [ADPS16b] and TLS 1.3 [Res18] protocols we analyze in Sections 4 and 5); due to space restrictions, we defer for further examples to [BFG19, Table 1].

For keyed primitives (both public-key and secret-key ones), the basic idea for the Break oracle is to hand to the adversary all secret keys which have been created in protocol executions so far. Since the adversary in our model can call the Break oracle multiple times it may also access subsequently generated keys. In order for Break to provide the necessary information, we make the key generation algorithm of a primitive explicit and have all honest parties invoke it when generating key material for this primitive. For example, any keys used for a MAC scheme $\mathcal{M} = (\mathsf{MKG}, \mathsf{MAC}, \mathsf{MVf})$ in honest sessions will be generated via the key generation algorithm MKG, with the challenger in the security game storing the output. This approach enables the challenger to return an exhaustive list of all secret keys of a primitive up to the point of breakdown when a Break query is asked.

In key exchange protocols it is common that keys for keyed primitives are not derived via an explicit key generation algorithm but, e.g., sampled at random or generated through a key derivation function. We implicitly treat such key derivations as a trivial key generation algorithm in our model, hence recording also such keys for exposure through a Break query.

For unkeyed primitives with a secret input, such as key derivation functions, we model a break of the output behavior by returning all outputs of evaluations so far. This means for example that the function is no longer unpredictable or pseudorandom. To capture this formally, we again assume that the challenger keeps a list of all function outputs generated by honest sessions, in order to provide the according list to the adversary in case of a Break query.

For public primitives like a hash function $\mathcal{H}$ and security properties like collision resistance we have to capture the increased capabilities of the adversary $\mathcal{A}$ after the breakdown differently. Here, regardless of whether $\mathcal{H}$ is modeled as a random oracle RO or considered in the standard model, the adversary $\mathcal{A}$ might

Primitive or Cryptographic Hardness Assumption (f)	**Algorithms**	**Security Assumption** (sec-prop)	Break **Response**
Asymmetric or Symmetric Encryption Scheme $\mathcal{E}$	$\mathcal{E} = (\mathsf{EKG}, \mathsf{Enc}, \mathsf{Dec})$	IND-CCA2 (indistinguishability under adaptive chosen ciphertext attack)	return all previous outputs (pk, sk) or sk for which $(pk, sk) \leftarrow \mathsf{EKG}$ or $sk \leftarrow \mathsf{EKG}$
Signature Scheme $\mathcal{S}$	$\mathcal{S} = (\mathsf{SKG}, \mathsf{Sig}, \mathsf{SVf})$	EUF-CMA (existential unforgeability under chosen message attack)	return all previous pairs (pk, sk) for which $(pk, sk) \leftarrow \mathsf{SKG}$
MAC Scheme $\mathcal{M}$	$\mathcal{M} = (\mathsf{MKG}, \mathsf{MAC}, \mathsf{MVf})$	EUF-CMA (existential unforgeability under chosen message attack)	return all previous values sk for which $sk \leftarrow \mathsf{MKG}$
Hash Function Family $\mathcal{H}$	$\mathcal{H} = (\mathsf{HKG}, \mathsf{Hash})$	STD-Coll-Res (standard-model collision resistance)	programmable access to Hash: After breakdown, $\mathcal{A}$ sets output of Hash queries on previously unseen values
	$\mathcal{H} = (\mathsf{HKG}, \mathsf{RO})$	RO-Coll-Res (random-oracle collision resistance)	programmable access to RO: After breakdown, $\mathcal{A}$ sets output of RO queries on previously unseen values
	$\mathcal{H} = (\mathsf{HKG}, \mathsf{Hash})$	Sec-Pre-Res (second preimage resistance)	programmable access to Hash: After breakdown, $\mathcal{A}$ can set output of Hash query on previously unseen value x' to y, where $y = H(x)$ for some previously seen value x
	$\mathcal{H} = (\mathsf{HKG}, \mathsf{RO})$	RO-Rand (random-oracle randomness)	return all previous s for which $s \leftarrow \mathsf{RO}(\cdot)$
	$\mathcal{H} = (\mathsf{HKG}, \mathsf{RO})$	RO-One-Way (random-oracle one-wayness)	return all previous pairs (x, s) for which $s \leftarrow \mathsf{RO}(x)$
Key Derivation Function KDF	KDF	KDF-sec (output pseudorandomness)	return all previous values k for which $k \leftarrow \mathsf{KDF}$
	KDF = RO	RO-Rand (random-oracle randomness)	return all previous k for which $k \leftarrow \mathsf{RO}(\cdot)$
	KDF = RO	RO-One-Way (random-oracle one-wayness)	return all previous pairs (k, x) for which $k \leftarrow \mathsf{RO}(x)$
Pseudorandom Function Family $\mathcal{P}$	$\mathcal{P} = (\mathsf{PKG}, \mathsf{PRF})$	PRF-sec (output pseudorandomness)	return all previous values k for which $k \leftarrow \mathsf{PKG}$
	$\mathcal{P} = (\mathsf{PKG}, \mathsf{RO})$	RO-Rand (random-oracle randomness)	return all previous s for which $s \leftarrow \mathsf{RO}(\cdot)$
	$\mathcal{P} = (\mathsf{PKG}, \mathsf{RO})$	RO-One-Way (random-oracle one-wayness)	return all previous pairs (x, s) s.t. $s \leftarrow \mathsf{RO}(x)$
Discrete Log Assumption	$\mathsf{GroupExp}(h, x) = h^x$ in multiplicative cyclic group $\mathbb{G} = \langle g \rangle$, $h \in \mathbb{G}$	Discrete Logarithm Problem	return all previous pairs (x, h^x) for which $h^x \leftarrow \mathsf{GroupExp}(h, x)$
Factoring Assumption	$\mathsf{GenModulus}(1^n) = (N, p, q)$ s.t. $N = p \cdot q$ where p, q are n-bit primes	Prime Factorization	return all previous tuples (N, p, q) for which $(N, p, q) \leftarrow \mathsf{GenModulus}(\cdot)$
Authenticated Key Exchange KE	two-party protocol KE, outputs session identifier sid and key K, and has transcript transcript	BR security	return all established keys K

Table 1: Potential Break oracle specifications.

be able to craft collisions after the break. We would model this by allowing $\mathcal{A}$ to program $\mathcal{H}$ globally on previously unseen inputs after the breakdown. More precisely, after the break, $\mathcal{A}$ answers all queries by honest sessions to the hash function $\mathcal{H}$ itself (but consistently with previous replies). If, on the other hand, we aim at modeling breakdown of the one-wayness of a random oracle, we instead hand the adversary all input-output pairs which honest parties have evaluated.

Finally, we can also treat the breakdown of interesting cryptographic assumptions for key exchange via the Break oracle. We illustrate this here by the discrete logarithm problem (DLP) and the factoring problem, which we treat similarly to public-key primitives. For the example of DLP, we mandate that honest sessions invoke a given algorithm GroupExp for group exponentiations, which then allows the challenger in the security game to provide the adversary with all secret exponents employed in honest sessions on a Break query. Note that for related cryptographic assumptions, the breakdown of one assumption can imply the breakdown of the other. For example, we can restrict our attention to DLP for Diffie–Hellman-style protocols, as (resilience against) a breakdown of DLP in particular implies (resilience against) the breakdown of other commonly used assumption like DDH and CDH.

We stress that Table 1 only gives (conservative) recommendations on how the Break oracle can be implemented for the most common primitives and hardness assumptions in the area of key exchange. Depending on the security properties required in a specific key-exchange setting, one may wish to specify different responses for the Break query. Likewise, weaker Break capabilities can be defined for demonstrating negative results, as we will do in our analysis of the TLS 1.3 PSK mode in Section 5.3. Again, this is easily possible in our model as the Break query itself is generic.

3.3 Modeling Rationale

Let us pause to briefly provide some further insight into the rationale behind our model for breakdown resilience in general and the Break oracle specifically.

BDR vs. Forward Secrecy. Breakdown resilience aims at a broader setting than forward secrecy, leading to a generic Break oracle. While both settings permit the exposure of long-term secrets after the tested session has accepted, breakdown resilience also needs to capture cryptographic weaknesses in primitives with ephemeral keys only or in unkeyed primitives.

To make this distinction even more explicit at this point, consider a KEM-based key exchange protocol [BCGP08] like the scheme based on the Kyber KEM [BDK+18]. In such designs a static KEM instance usually serves authentication purposes and an ephemeral KEM instance, based on the same hardness assumption, is used to establish the key and to provide forward secrecy. A breakdown of the underlying KEM assumption, however, would also reveal the secret keys of all past sessions. This demonstrates that not all effects of future compromises of keyed primitives can be captured through the notion of forward secrecy, let alone breakdowns of unkeyed primitives or assumptions.

Power of Break Capabilities. In Table 1, we give possible specifications for how to model the break capabilities of an adversary. The strong design choices there follow common cryptographic tradition to consider even weak attacks as successful and to then prove security against the strongest possible attacks. While less powerful definitions of break capabilities are possible, these weaker notions may give rise to a false sense of security as a breakdown may have more dire consequences than actually accounted for in the model.

Furthermore, we believe that defining breakdowns as a direct counterpart to the security notions often yields rather weak security notions. As an example, imagine the break of the commonly assumed existential unforgeability of signatures (EUF-CMA security). If one were to model EUF-CMA breaks as the weakest form of breaking the assumption, the adversary would only be provided with a forged signature on *some*

message m. A KE protocol with some structure in its messages would then easily achieve breakdown resilience with respect to such break (even in the strong BDR model), but any actual vulnerability of the signature scheme enabling forgeries on specific messages would not be covered by such weak security result.

Nevertheless, we deliberately choose to only give suggestions for the Break oracle specifications. While this allows to define different ways of how a certain primitive breaks down, such distinct breaks might indeed be meaningful for different settings. The only assumption made in the security model is hence that any break is devastating for the protocol (as of the Send query definition), except in the strong BDR model discussed in Section 6.

3.4 Breakdown-Resilient AKE Security Games

We are now ready to define the security notion of *breakdown resilience* (BDR) for an authenticated key exchange protocol. Extending the Bellare–Rogaway-like model from Section 2, we similarly divide the security properties into BDR-Match security and BDR key secrecy. Both security notions differ from the original Bellare–Rogaway-like notions by including the set of primitive breakdowns $\mathcal{F}_{\mathsf{BDR}}$ under consideration and the novel Break query as well as replacing the original Send oracle by the modified $\mathsf{Send}_{\mathsf{BDR}}$ version. The BDR-Match definition furthermore reflects that contributive identifiers must coincide in matching sessions but be distinct otherwise, while BDR key secrecy leverages the introduced contributive identifiers to exclude test sessions with semi-completed partners at the time of breakdown.

Definition 3.1 (BDR-Match Security)**.** *Let λ be the security parameter,* KE *a key exchange protocol, and $\mathcal{A}$ a PPT adversary interacting with* KE *via the queries* NewSession, $\mathsf{Send}_{\mathsf{BDR}}$, Reveal, Corrupt, *and* Break *in the following game:*

Setup. *The challenger generates long-term public/private-key pairs with certificates for each participant $U \in \mathcal{U}$.*

Query. *The adversary $\mathcal{A}$ receives the generated public keys and has access to the queries* NewSession, $\mathsf{Send}_{\mathsf{BDR}}$, Reveal, Corrupt, Test, *and* Break.

Stop. *At some point, the adversary stops with no output.*

Let $\mathcal{F}_{\mathsf{BDR}}$ be a set of cryptographic primitives and hardness assumptions the adversary can break in the model. We say that $\mathcal{A}$ wins the above game, denoted by $G^{\mathsf{BDR\text{-}Match}(\mathcal{F}_{\mathsf{BDR}})}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1$, if at least one of the following conditions holds:

1. *There exist two distinct sessions π and π' with $\pi.\mathsf{sid} = \pi'.\mathsf{sid} \neq \bot$, and $\pi.\mathsf{st}_{\mathsf{exec}}, \pi'.\mathsf{st}_{\mathsf{exec}} \neq \mathsf{rejected}$, but $\pi.\mathsf{K} \neq \pi'.\mathsf{K}$. (Different session keys in partnered sessions.)*
2. *There exist two distinct sessions π and π' such that $\pi.\mathsf{sid} = \pi'.\mathsf{sid} \neq \bot$, but $\pi.\mathsf{cid} \neq \pi'.\mathsf{cid}$ or $\pi.\mathsf{cid} = \pi'.\mathsf{cid} = \bot$. (Different or unset contributive identifiers in partnered sessions.)*
3. *There exist two sessions $\pi := \pi^{k}_{U,V}$ and $\pi' := \pi^{k'}_{V',U'}$ such that $\pi.\mathsf{sid} = \pi'.\mathsf{sid} \neq \bot$, $\pi.\mathsf{role} = \mathsf{initiator}$, and $\pi'.\mathsf{role} = \mathsf{responder}$, but $U \neq U'$ or $V \neq V'$. (Different intended partner in partnered sessions.)*
4. *There exist at least three sessions π, π', and π'' such that π, π', π'' are pairwise distinct, but $\pi.\mathsf{sid} = \pi'.\mathsf{sid}' = \pi''.\mathsf{sid} \neq \bot$ or $\pi.\mathsf{cid} = \pi'.\mathsf{cid}' = \pi''.\mathsf{cid} \neq \bot$. (More than two sessions share the same session or contributive identifier.)*

We say KE *is* BDR-Match-secure *for $\mathcal{F}_{\mathsf{BDR}}$ if for all PPT adversaries $\mathcal{A}$ the advantage function*

$$\mathsf{Adv}^{\mathsf{BDR\text{-}Match}(\mathcal{F}_{\mathsf{BDR}})}_{\mathsf{KE},\mathcal{A}} := \Pr\left[G^{\mathsf{BDR\text{-}Match}(\mathcal{F}_{\mathsf{BDR}})}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1\right]$$

is negligible in the security parameter λ.

Definition 3.2 (BDR Key Secrecy)**.** *Let λ be the security parameter,* KE *a key exchange protocol with key distribution $\mathcal{D}$, and $\mathcal{A}$ a PPT adversary interacting with* KE *via the queries* NewSession, $\mathsf{Send}_{\mathsf{BDR}}$, Reveal, Corrupt, Break, *and* Test *in the following game:*

Setup. *The challenger generates long-term public/private-key pairs for each participant $U \in \mathcal{U}$, chooses the test bit $b_{\mathsf{test}} \xleftarrow{\$} \{0,1\}$ at random and sets* $\mathsf{lost} \leftarrow \mathsf{false}$.

Query. *The adversary $\mathcal{A}$ receives the generated public keys and has access to the queries* NewSession, $\mathsf{Send}_{\mathsf{BDR}}$, Reveal, Corrupt, Test, *and* Break.

Guess. *At some point, $\mathcal{A}$ stops and outputs a guess b_{guess}.*

Finalize. *The challenger sets the lost flag to* $\mathsf{lost} \leftarrow \mathsf{true}$ *if at least one of the following conditions hold:*

1. *There exist two (not necessarily distinct) sessions π, π' such that $\pi.\mathsf{sid} = \pi'.\mathsf{sid}$, $\pi.\mathsf{st}_{\mathsf{key}} = \mathsf{revealed}$, and $\pi'.\mathsf{tested} = \mathsf{true}$. (Adversary has tested and revealed the key in a single session or in two partnered sessions.)*
2. *There exist two distinct sessions π, π' such that $\pi.\mathsf{tested} = \mathsf{true}$, $\pi.\mathsf{cid} = \pi'.\mathsf{cid}$, and $\pi'.\mathsf{st}^{\mathsf{bd}}_{\mathsf{exec}} = \mathsf{running}$. (Adversary has tested a session whose contributive partner session was running at the time of breakdown.)*

Let $\mathcal{F}_{\mathsf{BDR}}$ be a set of cryptographic primitives and hardness assumptions the adversary can break in the model. The adversary $\mathcal{A}$ wins the game, denoted by $G^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1$, if $b_{\mathsf{guess}} = b_{\mathsf{test}}$ and $\mathsf{lost} = \mathsf{false}$.

We say that KE *provides* BDR key secrecy *for $\mathcal{F}_{\mathsf{BDR}}$ with/without forward secrecy if for all PPT adversaries $\mathcal{A}$ the advantage function*

$$\mathsf{Adv}^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\mathsf{KE},\mathcal{A}}(\lambda) := \Pr\left[G^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\mathsf{KE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}$$

is negligible in the security parameter λ.

Definition 3.3 (Breakdown Resilience)**.** *We say a key exchange protocol* KE *is* breakdown resilient for $\mathcal{F}_{\mathsf{BDR}}$ *(with/without forward secrecy) if* KE *provides* BDR-Match *security and* BDR *key secrecy for $\mathcal{F}_{\mathsf{BDR}}$ (with/without forward secrecy), according to Definitions 3.1 and 3.2.*

3.5 Fundamental Properties

Since the model for breakdown resilience is a proper extension of the Bellare–Rogaway model for authenticated key exchange given in Section 2, breakdown resilience implies BR security.

Proposition 3.4. *If a key exchange protocol* KE *achieves breakdown resilience for some $\mathcal{F}_{\mathsf{BDR}}$ (incl. $\mathcal{F}_{\mathsf{BDR}} = \emptyset$) with/without forward secrecy according to Definition 3.3, then* KE *is also* BR*-secure with/without forward secrecyaccording to Definition 2.3.*

Proof. If the Break query is not asked by the adversary, the flag breakdown and the modification to the original Send query are essentially not touched and may thus be omitted. Likewise, the Finalize condition 2 in Definition 3.2 becomes void as $\mathsf{st}^{\mathsf{bd}}_{\mathsf{exec}} = \perp$ for all sessions. But then the models and in particular the Match security definition (modulo contributive identifiers) and the key secrecy definition for breakdown resilience and original BR security coincide. □

As mentioned earlier, it is often convenient to consider breakdown resilience for a stronger cryptographic hardness assumption than the one employed in a (non–breakdown-resilient) security proof, with DLP vs. DDH and CDH being a specific example. We hence make this relation more precise via the following proposition, which may prove useful when considering the breakdown of a cryptographic hardness assumption X whose breakdown implies the ability to break some other assumption Y. In our setting this means that one can provide the reply of the Break oracle for Y by the answer for X. We say that solving X implies solving Y.

Proposition 3.5. *Let Π be some protocol and let X and Y be some cryptographic hardness assumptions with $X \in \mathcal{F}_{\mathsf{BDR}}$, but $Y \notin \mathcal{F}_{\mathsf{BDR}}$. Assume that solving X implies solving Y. Then, if Π is breakdown resilient for $\mathcal{F}_{\mathsf{BDR}}$, then Π is also breakdown resilient for $\mathcal{F}'_{\mathsf{BDR}} = \mathcal{F}_{\mathsf{BDR}} \cup \{Y\}$.*

Proof. We can directly simulate the Break query for $\mathcal{F}'_{\mathsf{BDR}}$ via a Break query for $\mathcal{F}_{\mathsf{BDR}}$, since the Break response for X allows to provide the response for Y. □

4 NewHope

As a first application of our new security model, we analyze the breakdown resilience of an authenticated variant of the NewHope scheme. NewHope is a post-quantum secure key exchange protocol originally introduced in 2016 by Alkim et al [ADPS16b]. It has gained widespread attention, not least because of its experimental deployment in Google Chrome Canary [Bra16]. The same year, a simpler encryption-based version NewHope-Simple was introduced [ADPS16a]. Contrary to the previous reconciliation-based design this variant is based on encryption of the shared key and constitutes the basis for the candidate key encapsulation schemes [AAB+19] that were submitted to the NIST Post-Quantum Cryptography standardization process [NIS17] and have made it to the second round of the process. The post-quantum security of all NewHope schemes is based on the ring learning with errors problem (RLWE), which states that $as + e$ for secret s, public a, and small error e is indistinguishable from random.

In our analysis, we consider an authenticated version of the passively secure KEM provided in the NIST candidate submission NewHope-Nist [AAB+19]. For illustrative purposes, the description of Auth-NewHope in Figure 2 has been divided according to the two phases of the unauthenticated NewHope-Nist key encapsulation and the ensuing SIGMA-style authentication. One can, of course, condense the entire protocol in a three-move key exchange by having Alice send r_A in the first step and Bob attach B, r_B, σ_B, τ_b to its last message in the NewHope step. This does not affect our security proof of breakdown resilience. For details on the key encapsulation mechanism and its IND-CPA security, we refer the interested reader to the original specification in [AAB+19, Sec.1.2].

4.1 Cryptographic Assumptions

Auth-NewHope relies on the following cryptographic primitives and hardness assumptions: IND-CPAsecurity of the key encapsulation mechanism KEM, pseudorandomness of the key derivation function KDF, and existential unforgeability of the signature scheme $\mathcal{S}$ and MAC scheme $\mathcal{M}$. The definition for the standard cryptographic assumptions, such as the unforgeability of signatures and KDF security can be found in Appendix A. Before we can define the decisional Ring-LWE problem formally, we first need to fix some commonly used notation.

Notation. Let $\mathcal{R} = \mathbb{Z}[X]/X^n + 1$ for $n = 2^m, m \geq 0$ be the ring of integers of the $2n$-th cyclotomic number field. For q an integer, define $\mathcal{R}_q$ to be the ring $\mathcal{R}/q\mathcal{R} \cong \mathbb{Z}_q[X]/(X^n + 1)$. By $x \xleftarrow{\$} \chi$ we denote the sampling of x from a probability distribution χ. Let $\mathcal{U}(S)$ denote the uniform distribution over some set S.

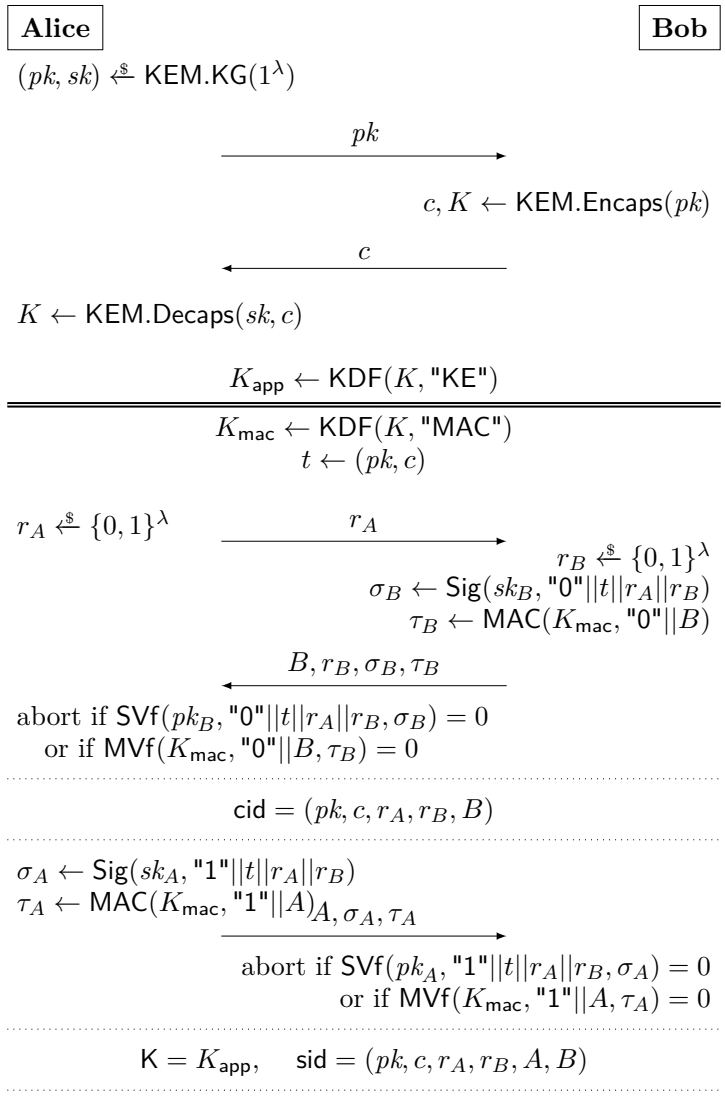


Figure 2: The AUTH-NEWHOPE protocol with the IND-CPA-secure KEM KEM from NEWHOPE-NIST above the double line and SIGMA-style authentication.

With this, we can state the decisional version of the Ring-LWE problem upon which the security of the NEWHOPE schemes is based:

Definition 4.1 (DRLWE Problem)**.** *Let $n, q, \mathcal{R}, \mathcal{R}_q$ be defined as above. Let χ be some probability distribution over $\mathcal{R}_q$. The decisional Ring-LWE problem* DRLWE *states that given (a, b) $\mathcal{R}_q \times \mathcal{R}_q$, it is hard to decide if $b = as + e$ for $s, e \xleftarrow{\$} \chi$ small ring elements or if b is a uniform ring element $b \xleftarrow{\$} \mathcal{U}(R_q)$. More precisely, the distinguishing advantage for $b = as + e$ and $b' \xleftarrow{\$} \mathcal{R}_q$ is given by*

$$\mathsf{Adv}^{\mathsf{DRLWE}}_{q,n,\chi,\mathcal{A}} := \left|\Pr[\mathcal{A}(a, b) = 1] - \Pr[\mathcal{A}(a, b') = 1]\right|.$$

4.2 Breakdown Resilience of Auth-NewHope

In the following, we show that AUTH-NEWHOPE achieves breakdown resilience for $\mathcal{F}_{\mathsf{BDR}} = \{(\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathcal{M}, \mathsf{EUF\text{-}CMA})\}$ with forward secrecy by establishing the corresponding BDR-Match security and BDR

key secrecy. Note that $\mathcal{F}_{\mathsf{BDR}}$ neither contains the IND-CPA security of KEM nor the key derivation function KDF, as a break of any of these makes key secrecy impossible to achieve.

Theorem 4.2 ($\mathsf{BDR\text{-}Match}$ security of Auth-NewHope)**.** *Let* $\mathcal{F}_{\mathsf{BDR}} = \{(\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathcal{M}, \mathsf{EUF\text{-}CMA})\}$. *Then* Auth-NewHope *is* $\mathsf{BDR\text{-}Match}$*-secure for* $\mathcal{F}_{\mathsf{BDR}}$*. For any efficient adversary* $\mathcal{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{BDR\text{-}Match}(\mathcal{F}_{\mathsf{BDR}})}_{\text{A-NH},\mathcal{A}} \leq n_s^2 \cdot \min\left\{\mathsf{coll}\text{-}pk, \mathsf{coll}\text{-}c, 2^{-|\mathsf{nonce}|}\right\},$$

where n_s *is the maximum number of sessions,* coll*-*pk *is the probability that the same public key is generated twice,* coll*-*c *is the probability that the same ciphertext is encapsulated twice, and* $|\mathsf{nonce}|$ *is the bit-length of the nonces* r_A *and* r_B*.*

Proof. In order to achieve $\mathsf{BDR\text{-}Match}$ Security, we need to show that the four conditions are satisfied (cf. Definition 3.1). Recall that the session identifiers are defined as $\mathsf{sid} = (pk, c, r_A, r_B, A, B)$, containing public information only, and that the contributive identifiers are set as $\mathsf{cid} = (pk, c, r_A, r_B, B)$.
Ad (1). Since the session identifier already determines all inputs to the key derivation function KDF, partnered sessions necessarily also agree on the session key.
Ad (2). Since cid contains all entries in sid except for A's identity, it trivially holds that same session identifiers imply identical contributive identifiers.
Ad (3). Both identifiers A and B are comprised in the session identifier. Thus, agreement on the session identifier implies agreement on the intended partner's identity.
Ad (4). In order for three sessions sharing the same session or contributive identifier, with respect to two honest sessions, a third honest session must have, depending on its role, a collision in either the public key pk and r_A or the ciphertext c and r_B. This will only happen with probability at most $\min\left\{\mathsf{coll}\text{-}pk, \mathsf{coll}\text{-}c, 2^{-|\mathsf{nonce}|}\right\}$. There are at most n_s^2 many combinations of the initial two sessions, where n_s denotes the maximum number of protocol executions, arriving at the final bound. □

Theorem 4.3 (BDR key secrecy of Auth-NewHope)**.**
Let $\mathcal{F}_{\mathsf{BDR}} = \{(\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathcal{M}, \mathsf{EUF\text{-}CMA})\}$. *Then* Auth-NewHope *achieves breakdown-resilient key secrecy for* $\mathcal{F}_{\mathsf{BDR}}$ *with forward secrecy. More precisely, for any efficient, adversary* $\mathcal{A}$ *there exist efficient adversaries* $\mathcal{B}_1, \ldots, \mathcal{B}_4$ *such that*

$$\mathsf{Adv}^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\text{A-NH},\mathcal{A}} \leq n_s^2 \cdot 2^{-|\mathsf{nonce}|} + n_s \cdot \left(n_u \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{B}_1} + n_s \cdot \left(\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{B}_2} + \mathsf{Adv}^{\mathsf{KDF\text{-}sec}}_{\mathsf{KDF},\mathcal{B}_3} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{M},\mathcal{B}_4}\right)\right),$$

where n_s *is the maximum number of sessions,* n_u *is the maximum number of users, and* $|\mathsf{nonce}|$ *is the bit-length of the nonces.*

Proof. For the proof, we proceed in a sequence of games, bounding the difference in the adversary's advantage introduced in each step, until we reach a game where the adversary cannot win anymore.

Game 0. The original BDR key secrecy game $G^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\text{A-NH},\mathcal{A}}$.

Game 1. We abort the game if there are two sessions of honest parties which generate the same nonce r_A resp. r_B. The probability of this happening is at most $n_s \cdot 2^{-|\mathsf{nonce}|}$, where n_s denotes the maximum number of sessions, since nonces in any n_s^2 possible pair of sessions are both chosen at random.

We thus have

$$\mathsf{Adv}^{G_0}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_1}_{\text{A-NH},\mathcal{A}} + n_s^2 \cdot 2^{-|\mathsf{nonce}|}.$$

Game 2. We proceed by guessing the tested session, thus reducing our reduction's advantage by a factor of at most $\frac{1}{n_s}$:

$$\mathsf{Adv}^{G_1}_{\text{A-NH},\mathcal{A}} \leq n_s \cdot \mathsf{Adv}^{G_2}_{\text{A-NH},\mathcal{A}}.$$

In the following, this allows us to know the tested session, denoted by π^*, in advance. Observe that π^* must have accepted (and received all incoming messages) prior to the first Break query issued by $\mathcal{A}$ in order for the latter to win, as otherwise its session key would be considered revealed.

Game 3. Next, we abort the game if the tested session π^*, run by some party P (where P may be Alice or Bob), obtains a valid signature σ_Q on $(\texttt{"b"}||t||r_A||r_B)$ which has not been signed by an honest party Q at this point. Recall that this message must have been received prior to any Break query, in particular before a breakdown of $\mathcal{S}$, as otherwise π^* would be considered revealed and could not be tested. Furthermore, long-term secrets of the involved parties may not be corrupted before the test session has accepted. Forward secrecy is achieved since a subsequent $\mathsf{Corrupt}$ query on the owner of the test session π^* (or its intended partner) does not contradict the fact that π^* receives an honestly generated signature according to this game hop.

We now show that the probability of an abort happening for this reason can be bounded by the success probability of the following reduction $\mathcal{B}_1$ against the unforgeability of the signature scheme $\mathcal{S}$. The reduction $\mathcal{B}_1$ receives a public key pk^* as challenge and guesses the party Q under whose name the forgery obtained in π^* is issued. It creates all parameters for the key exchange as specified, except for setting $pk_Q = pk^*$. Any signature creation of Q is performed through a query to the signature oracle, all other steps can be carried out by $\mathcal{B}_1$ itself. If at some point the tested session π^* accepts a signature for a previously unsigned message, then $\mathcal{B}_1$ outputs this message-signature pair as a forgery. In this case, since the nonces are unique and the valid signature has not been created by an honest party before, party Q cannot have signed $(\texttt{"b"}||t||r_A||r_B)$ earlier, only $(\texttt{"b'"}||t||r_A||r_B)$ for $\mathsf{b}' = 1 - \mathsf{b}$ (if at all). With probability $\frac{1}{n_u}$, where n_u is the total number of users, our reduction predicts the party Q correctly, such that we have

$$\mathsf{Adv}^{G_2}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_3}_{\text{A-NH},\mathcal{A}} + n_u \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{B}_1}.$$

Game 4. In the next step, we guess the honest session π^*_{a} of party Q which has sent the valid signature σ_Q received by π^* in Game 3 and abort if we guessed incorrectly. This session is unique because the nonces are unique and there must be such a session which creates the signature according to the previous game. Still, the session may not necessarily be partnered with the test session, but must (at least) have the same contributive identifier, such that we call this session *associated*.

Changing the game like this reduces the adversary's advantage by a factor of at most $\frac{1}{n_s}$, with n_s again being the maximum number of sessions. Hence, we have

$$\mathsf{Adv}^{G_3}_{\text{A-NH},\mathcal{A}} \leq n_s \cdot \mathsf{Adv}^{G_4}_{\text{A-NH},\mathcal{A}}.$$

Game 5. As the next step, we replace the value K in the test session (and its associated session π^*_{a}) by a uniformly random value $\widetilde{K}$ of equal length.. If the adversary $\mathcal{A}$ can distinguish Game 5 from Game 4, then there exists an adversary $\mathcal{B}_2$ that can break the IND-CPA security of the key encapsulation mechanism KEM as follows.

Algorithm $\mathcal{B}_2$ obtains its challenge public key, ciphertext and key pk, c^*, K^*, where either $(c^*, K^*) \xleftarrow{\$} \mathsf{KEM.Encaps}(pk)$ or K^* is a random element from the key space. $\mathcal{B}_2$ simulates the environment for $\mathcal{A}$ by creating all long-term keys of participants as specified and initializing $\mathcal{A}$ with the corresponding public

keys of participants. This ensures in particular, that $\mathcal{B}_2$ can answer all NewSession and Corrupt queries of the adversary. Furthermore, $\mathcal{B}_2$ can execute all Send requests by $\mathcal{A}$ for sessions $\pi \neq \pi^*, \pi^*_{\mathsf{a}}$. For π^* and π^*_{a}, $\mathcal{B}_2$ uses its challenge pk and c^* for the first two message flows. The session key K_{app} and the MAC key K_{mac} are computed as the KDF keyed with K^* and the respective label. $\mathcal{B}_2$ can also answer all Reveal for sessions that are not the Test session or its associated session. For the π^* and π^*_{a}, $\mathcal{A}$ will not query Reveal since this would cause it to trivially lose the game.

Once $\mathcal{A}$ queries Test on π^*, $\mathcal{B}_2$ computes the challenge key for $\mathcal{A}$ as $\mathsf{K} \leftarrow \mathsf{KDF}(K^*, \text{"KE"})$,i.e., when computing the keys K_{app} and K_{mac} in the two sessions, the given value K^* is used instead as input to the key derivation function KDF. At some point, $\mathcal{A}$ terminates and outputs a guess bit b_{guess}. Upon this, $\mathcal{B}_2$ also terminates and outputs the same b_{guess}.

If K^* is genuine, then the simulation above is as in Game 4. If K^* is random, $\mathcal{B}_2$ simulates Game 5. Hence, if the efficient adversary $\mathcal{A}$ can distinguish the two games with non-negligible advantage, then $\mathcal{B}_2$ can distinguish real from random keys in key encapsulation mechanisms efficiently with non-negligible advantage. It follows that

$$\mathsf{Adv}^{G_4}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_5}_{\text{A-NH},\mathcal{A}} + \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{B}_2}.$$

Since (KEM, IND-CPA) is not part of $\mathcal{F}_{\mathsf{BDR}}$, this bound especially holds in the BDR scenario..

Game 6. Next, we replace the session key $\mathsf{K} = K_{\mathsf{app}}$ and the MAC key K_{mac} by uniformly random values $\widetilde{K_{\mathsf{app}}}$ and $\widetilde{K_{\mathsf{mac}}}$ in π^* and π^*_{a}. Distinguishing Game 6 and Game 5 by $\mathcal{A}$ would immediately imply the existence of an efficient adversary $\mathcal{B}_3$ that breaks the pseudorandomness of KDF with non-negligible advantage. For this, $\mathcal{B}_3$ simply replaces KDF executions keyed with $\widetilde{w}$ by oracle calls in the pseudorandomness game, simulating one of the two games depending on the oracle response. Thus, we have

$$\mathsf{Adv}^{G_5}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_6}_{\text{A-NH},\mathcal{A}} + \mathsf{Adv}^{\mathsf{KDF\text{-}sec}}_{\mathsf{KDF},\mathcal{B}_3}.$$

As $(\mathsf{KDF}, \cdot) \notin \mathcal{F}_{\mathsf{BDR}}$, this bound again particularly holds in the BDR scenario.

There are now four possibilities for the status of the associated session π^*_{a}. First, at the point of the breakdown query, the associated session had not accepted yet, i.e., if π^*_{a} owned by Q is a Bob instance and waited for the final authentication message. But then π^*_{a}'s state was running and its contributive identifier was, and is, identical to the one in P's session π^*, since the signature is over the entries in cid and Q knows resp. sent its identifier. This however means that the adversary is not allowed to test the session it has actually tested, by definition of a successful attack.

If the associated session π^*_{a} had already finished upon the breakdown query, then it either is partnered with the test session (and thus cannot be revealed), or rejected (in which case it does not hold a session key), or it has accepted but is not partnered with the test session. The latter case would mean that the adversary would be allowed to safely reveal the session key of the associated but unpartnered session and could break key secrecy. Yet, this would lead to a contradiction of the unforgeability of the MAC, as we discuss next.

Game 7. As the next change, we abort the game if the associated session π^*_{a} of party Q accepts before the breakdown query with a session identifier $\pi^*_{\mathsf{a}}.\mathsf{sid} \neq \perp$ which does not equal $\pi^*.\mathsf{sid}$. This can only happen if the adversary is able to make π^*_{a} obtain a valid signature σ_R and MAC τ_R for some identity $R \neq P$ since all entries except for the peer identity of $\pi^*_{\mathsf{a}}.\mathsf{sid}$ are already fixed at this point. We assume that the associated session has already accepted and that no Break query has occurred yet. In particular, while the adversary may be able to sign under a corrupt party's identifier R for which the adversary may know the signing key due to a Corrupt query, the MAC scheme, on the other hand, must still be secure. Furthermore, the

MAC tag depends on the key K_{mac} shared between the honest parties P and Q and includes the sender's identity.

Similarly to Game 3, the probability of an abort happening for this reason can be bounded by the success probability of an adversary $\mathcal{B}_4$ against the unforgeability of the MAC scheme $\mathcal{M}$. That is, since we have already replaced the key K_{mac} by an independent random value, we can use an external MAC oracle for an unknown key in a simulation instead, and use oracle queries to create the MACs for $\texttt{"b"}||P$ and $\texttt{"b'"}||Q$ for $\mathsf{b}' = 1 - \mathsf{b}$ as required in the test session and its associated session. It follows that a valid MAC τ_R for $\texttt{"b"}||R$ created by the adversary for identity $R \neq P$ in the associated session constitutes a successful forgery for a fresh message. We have

$$\mathsf{Adv}^{G_6}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_7}_{\text{A-NH},\mathcal{A}} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{M},\mathcal{B}_4}.$$

To complete the proof we note that the adversary expects the challenge value $\mathcal{K}$ to be either a uniformly random string ($b_{\mathsf{test}} = 0$) or to be the output of $\mathsf{KDF}(w, \text{"KE"})$ ($b_{\mathsf{test}} = 1$). At this point, both cases $b_{\mathsf{test}} = 0$ and $b_{\mathsf{test}} = 1$ are indistinguishable for $\mathcal{A}$ since both keys are of equal length and are drawn independently and uniformly at random. Furthermore, the session key in the associated session (which coincides with the now random key $\mathcal{K}$ in case of $b_{\mathsf{test}} = 1$ and is independent of $\mathcal{K}$ for $b_{\mathsf{test}} = 0$) cannot be revealed, because that session is either partnered or held the same contributive identifier upon breakdown. Thus $\mathcal{A}$ cannot learn any information about the bit b_{test}. The only strategy for $\mathcal{A}$ is to guess and thus we have the final bound:

$$\mathsf{Adv}^{G_7}_{\text{A-NH},\mathcal{A}} \leq 0.$$

□

Proof sketch. For the proof, we proceed in a sequence of games, bounding the difference in the adversary's advantage introduced in each step, until we reach a game where the adversary cannot win anymore.

Game 1. We first exclude collisions in the nonces r_A, r_B by aborting if two honest sessions generate the same nonce. We thus have $\mathsf{Adv}^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_1}_{\text{A-NH},\mathcal{A}} + n_s^2 \cdot 2^{-|\mathsf{nonce}|}$.

Game 2. We proceed by guessing the tested session, reducing our advantage by a factor of at most $\frac{1}{n_s}$, i.e., $\mathsf{Adv}^{G_1}_{\text{A-NH},\mathcal{A}} \leq n_s \cdot \mathsf{Adv}^{G_2}_{\text{A-NH},\mathcal{A}}$. This allows us to know the tested session, denoted by π^*, in advance.

Game 3. Next, we abort the game if the tested session π^*, run by some party P, obtains a valid signature σ_Q on $(\texttt{"b"}||t||r_A||r_B)$ which has not been signed by an honest party Q at this point. Note that this message must have been received prior to any Break query, in particular before a breakdown of $\mathcal{S}$ as otherwise P's session would be considered revealed and could not be tested. This step is bound by a reduction $\mathcal{B}_1$ to the unforgeability of signatures. With probability $\frac{1}{n_u}$, where n_u is the total number of users, our reduction predicts the party Q correctly, such that we have $\mathsf{Adv}^{G_2}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_3}_{\text{A-NH},\mathcal{A}} + n_u \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{B}_1}$.

Game 4. We now guess the honest session π^*_{a} of party Q which has sent the valid signature σ_Q received by π^* in Game 3 and abort if we guessed incorrectly; such a session must exist uniquely due to Games 1 and 3. Still, the session may not necessarily be partnered with the test session, but must (at least) have the same contributive identifier, such that we call this session *associated*. Hence, we have $\mathsf{Adv}^{G_3}_{\text{A-NH},\mathcal{A}} \leq n_s \cdot \mathsf{Adv}^{G_4}_{\text{A-NH},\mathcal{A}}$, where n_s denotes the maximal number of sessions.

Game 5. Next, we replace the value K in π^* (and its associated session π^*_{a}) by a uniformly random value $\widetilde{K}$ of equal length. We bound the distinguishing advantage by the advantage of a reduction $\mathcal{B}_2$ against the

IND-CPA security of the NewHope KEM: $\mathsf{Adv}^{G_4}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_5}_{\text{A-NH},\mathcal{A}} + \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{B}_2}$. Since $(\mathsf{KEM}, \text{IND-CPA})$ is not part of $\mathcal{F}_{\mathsf{BDR}}$, this bound especially holds in the BDR scenario.

Game 6. We now replace the session key $\mathsf{K} = K_{\mathsf{app}}$ and the MAC key K_{mac} by uniformly random values in π^* and π^*_{a}. Distinguishing Game 6 and Game 5 can be reduced to an efficient adversary $\mathcal{B}_3$ breaking the pseudorandomness of KDF. Thus, we have $\mathsf{Adv}^{G_5}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_6}_{\text{A-NH},\mathcal{A}} + \mathsf{Adv}^{\mathsf{KDF\text{-}sec}}_{\mathsf{KDF},\mathcal{B}_3}$. As $(\mathsf{KDF}, \cdot) \notin \mathcal{F}_{\mathsf{BDR}}$, this bound again particularly holds in the BDR scenario.

As the last step we need to ensure that the adversary cannot make the associated session accept under a different session identifier *before the breakdown* (enabling a Reveal query to it), which we do by showing through the next game hop that this would imply a MAC forgery. If the associated session would accept later, it being contributively partnered with the test session at the point of breakdown prohibits the test query.

Game 7. As the final change, we abort if the associated session π^*_{a} accepts before the breakdown with a session identifier $\pi^*_{\mathsf{a}}.\mathsf{sid} \neq \bot$ which does not equal $\pi^*.\mathsf{sid}$. This can only happen if the adversary is able to make π^*_{a} obtain a valid signature σ_R and MAC τ_R for some identity $R \neq P$; the latter constitutes a valid MAC forgery usable in a reduction $\mathcal{B}_4$ to the MAC schemes EUF-CMA security. We have $\mathsf{Adv}^{G_6}_{\text{A-NH},\mathcal{A}} \leq \mathsf{Adv}^{G_7}_{\text{A-NH},\mathcal{A}} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{M},\mathcal{B}_5}$.

To complete the proof we note that the adversary expects the challenge key to be either a uniformly random string or to be the output of $\mathsf{KDF}(K, \text{"KE"})$. Here, both cases are indistinguishable since both keys are drawn independently and uniformly at random and so the adversary cannot do better than guessing: $\mathsf{Adv}^{G_7}_{\text{A-NH},\mathcal{A}} \leq 0$. □

Remark. It may be surprising at first that the unforgeability of the signature and MAC scheme enter into the security bound of Theorem 4.3 although the signature scheme $\mathcal{S}$ as well as the MAC scheme $\mathcal{M}$ are afflicted by the breakdown. However, both the valid signature obtained by π^* in Game 3 as well as the MAC tag in Game 7 must necessarily have been created before a breakdown had occurred. Thus, both unforgeability assumptions still hold at the respective points in time.

5 TLS 1.3

We now turn towards the second protocol for our exemplary breakdown resilience analysis, the Transport Layer Security (TLS) protocol in its latest version 1.3 [Res18]. Although not designed with breakdown resilience as a security goal in mind, the TLS 1.3 key exchange (or "handshake") achieves resilience against the breakdown of some of its cryptographic components, as we will see.

TLS 1.3 specifies four different handshake modes: a full Diffie–Hellman-based handshake (referred to as (EC)DHE mode), a resumption-style pre-shared key mode (PSK), a PSK mode combined with a Diffie–Hellman exchange (PSK-(EC)DHE), and a low-latency, zero round-trip (0-RTT) mode based on the PSK modes. Providing a full key exchange security analysis of these modes is beyond the scope of this work; for this we refer to prior analyses [DFGS15, DFGS16, CHSvdM16, FGSW16, LXZ+16, Kra16, FG17, BBK17, CHH+17] and also to [PvdM16] for a review of the standardization process. In our analysis, we focus on the full/(EC)DHE and PSK(-only) handshake modes, which suffice to demonstrate some essential breakdown-resilience properties of TLS 1.3.

Interestingly, despite both handshake modes following the same overall protocol structure, the (EC)DHE and PSK modes differ in the provided breakdown resilience. More precisely, the (EC)DHE mode offers resilience against breakdown of the authentication signature and MAC schemes' unforgeability as well as collision resistance of the hash function used to compute hashed transcript values, which is consistent with

the high-level expectations from the protocol design. The PSK-only handshake in contrast does not provide the same resilience against a hash function breakdown, the reason essentially being that transcripts are hashed before being used in the key derivation. As we will discuss, our analysis hence exhibits how seemingly minor technical design choices (even from a cryptographic point of view) can have a noticeable impact on the breakdown resilience of a key exchange protocol.

5.1 The TLS 1.3 Handshake Protocol

As we analyze the breakdown resilience of the TLS 1.3 (EC)DHE and PSK handshake modes, we accordingly limit the presentation of the TLS 1.3 handshake in the following to these modes. In order to focus attention on the breakdown resilience properties, we furthermore restrict ourselves to the security of the main application data key established in a mutually authenticated TLS 1.3 handshake, also omitting more advanced aspects like 0-RTT and 0.5-RTT key establishment and post-handshake messages. We note that our security model for breakdown resilience can in principle be extended to the setting of multi-stage key exchange protocols [FG14] in order to capture breakdown resilience for the multiple keys derived in TLS 1.3 with varying authentication properties (see also [DFGS15, DFGS16]).

5.1.1 Full/(EC)DHE mode

We begin with explaining the full handshake mode based on (elliptic-curve) ephemeral Diffie–Hellman ((EC)DHE) key exchange. Figure 3 shows the TLS 1.3 handshake protocol flow; messages and computations marked with $[\dots]^{\diamond}$ are only included in the PSK-based handshake mode and can be ignored for now.

The protocol begins with client and server exchanging random nonces r_c and r_s and ephemeral Diffie–Hellman shares g^x and g^y within the `ClientHello` resp. `ServerHello` and accompanying `KeyShare` extension messages.[4] Both sides then derive an intermediate handshake traffic key tk_{hs}, consisting of client- and server-side sending keys tk^c_{hs} and tk^s_{hs}. This key is derived from the shared Diffie–Hellman value $\mathrm{DHE} = g^{xy}$ via an intermediate handshake secret HS, using the HKDF key derivation function [Kra10] in an extract-then-expand paradigm.[5]

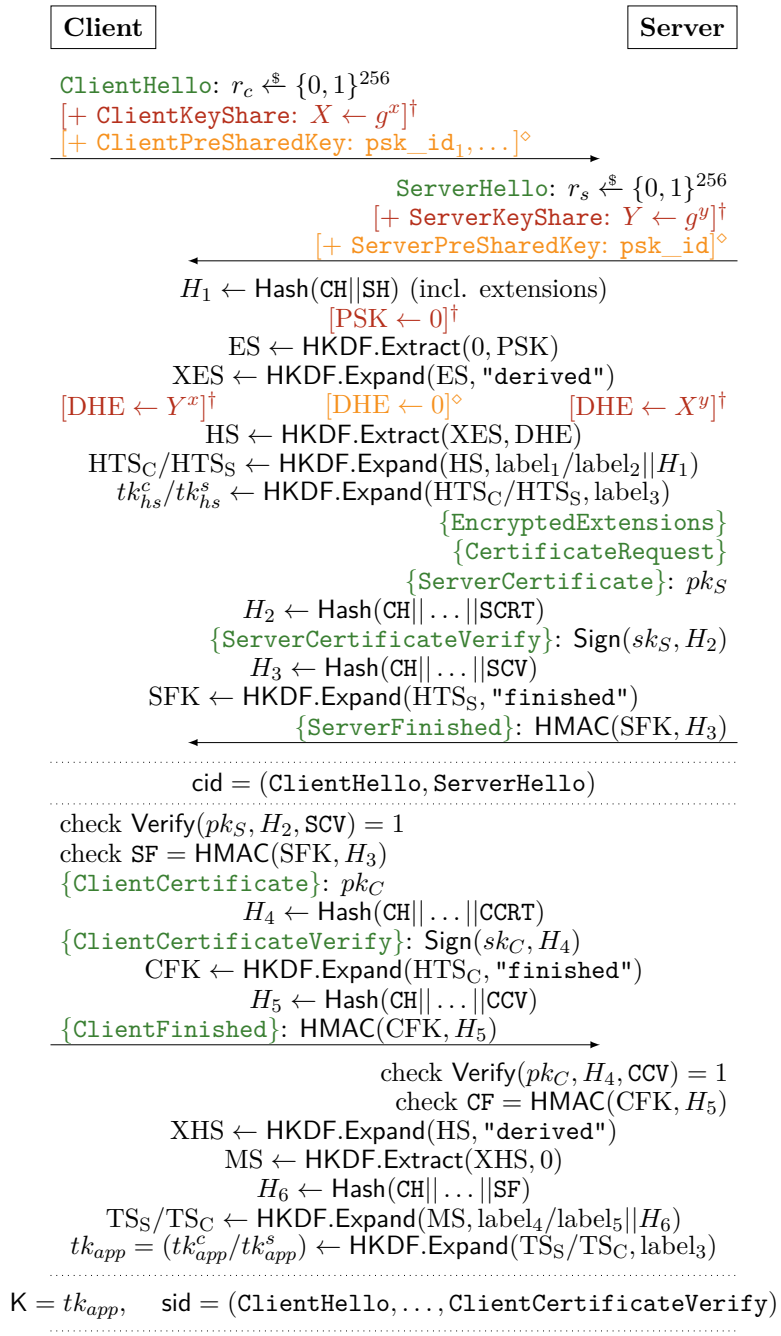
The remaining handshake is encrypted under tk_{hs}. For authentication, first the server and then the client send a certificate on their public key (within the `Certificate` messages), a signature over the communication transcript up to this point under the corresponding secret key (in `CertificateVerify`), and a `Finished` message containing a MAC over the transcript so far. Finally, both sides derive a master secret MS via $\mathsf{HKDF.Extract}$ and then expand from it the main application traffic key tk_{app} (again with server- and client-side component tk^c_{app} and tk^s_{app}) as the session key K.

5.1.2 PSK mode

In the pre-shared key (PSK) handshake mode, client and server agree on an (identifier for a) previously established shared secret key within the `PreSharedKey` messages. This pre-shared secret PSK enters the key derivation in an HKDF extract-then-expand step prior to deriving the handshake secret HS. Optionally,

[4] We also use abbreviated names for the TLS 1.3 messages exchange, e.g., `CH` for `ClientHello`, `CKS` for `ClientKeyShare`, etc.

[5] We adopt the following common notation for the two HKDF functions, both based on HMAC [BCK96]: $\mathsf{HKDF.Extract}(XTS, SKM)$ on input an extractor salt XTS and source key material SKM outputs a pseudorandom key PRK. $\mathsf{HKDF.Expand}(PRK, CTXinfo)$ on input a pseudorandom key PRK and context information $CTXinfo$ outputs some key material KM (we omit the third output-length parameter in Expand and assume it to be fixed to $L = \lambda$ for our security parameter λ).



Protocol flow legend	
MSG: Y	TLS 1.3 message MSG containing Y
+ MSG	message sent as extension within previous message
{MSG}	message MSG AEAD-encrypted with tk^c_{hs}/tk^s_{hs}
[...]†	message/computation only when including DHE
[...]◇	message/computation only when including PSK
a/b	alternative usage of a or b in analogous computation

Figure 3: The TLS 1.3 [Res18] handshake protocol (in full/(EC)DHE, PSK, and PSK-(EC)DHE mode).

both sides can also send Diffie–Hellman shares (within `KeyShare` messages) to be included in the key derivation; this variant constitutes the PSK-(EC)DHE mode.

Both in PSK-only and PSK-(EC)DHE mode, authentication relies on the pre-shared key only through the `Finished` messages, i.e., no certificates and signatures are exchanged and, accordingly, the messages `CertificateRequest`, `Certificate`, and `CertificateVerify` (from both sides) are omitted.

5.2 Breakdown Resilience of the TLS 1.3 (EC)DHE Handshake

The TLS 1.3 (EC)DHE handshake security relies on the following cryptographic primitives and hardness assumptions: hardness of Diffie–Hellman-type assumptions in the employed group $\mathbb{G}$, collision resistance of the hash function Hash for hashing the transcripts, pseudorandomness of the key derivation function HKDF, and unforgeability of the signature scheme $\mathcal{S}$ and of the MAC scheme HMAC.

We cannot hope for breakdown resilience for the Diffie–Hellman assumptions on $\mathbb{G}$ (as they might allow an adversary to recover the secrecy source g^{xy} of earlier handshakes) or pseudorandomness of HKDF (as non-pseudorandom output may enable an adversary to distinguish the session key from a random string). As we will show next, the TLS 1.3 (EC)DHE handshake however does achieve resilience against breakdown of the hash function, signature scheme, and MAC, ensuring security of completed sessions even in case these core primitives break. More precisely, we consider resilience against breakdown of the collision resistance of the hash function Hash (which we model as a standard-model hash function) as well as existential unforgeability of the signature scheme $\mathcal{S}$ and MAC scheme HMAC, i.e., breakdown resilience for $\mathcal{F}_{\mathsf{BDR}} = \{(\mathsf{Hash}, \mathsf{STD\text{-}Coll\text{-}Res}), (\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathsf{HMAC}, \mathsf{EUF\text{-}CMA})\}$.[6]

In the following, we establish breakdown resilience of the TLS 1.3 (EC)DHE handshake for $\mathcal{F}_{\mathsf{BDR}}$ (with forward secrecy) through the corresponding $\mathsf{BDR\text{-}Match}$ security and BDR key secrecy.

Theorem 5.1 ($\mathsf{BDR\text{-}Match}$ security of $\mathsf{TLS\text{-}(EC)DHE}$)**.** *The TLS 1.3 (EC)DHE handshake* $\mathsf{TLS\text{-}(EC)DHE}$ *is* $\mathsf{BDR\text{-}Match}$*-secure for* $\mathcal{F}_{\mathsf{BDR}} = \{(\mathsf{Hash}, \mathsf{STD\text{-}Coll\text{-}Res}), (\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathsf{HMAC}, \mathsf{EUF\text{-}CMA})\}$*. For any efficient adversary* $\mathcal{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{BDR\text{-}Match}(\mathcal{F}_{\mathsf{BDR}})}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq n_s^2 \cdot 1/q \cdot 2^{-|\mathsf{nonce}|},$$

where n_s *is the maximum number of sessions,* q *is the Diffie–Hellman element group order, and* $|\mathsf{nonce}| = 256$ *is the bit-length of the nonces* r_c *and* r_s*.*

Proof. We need to show that the four conditions for $\mathsf{BDR\text{-}Match}$ security (cf. Definition 3.1) are satisfied.
Ad (1). Sessions accepting with the same session identifier also derive the same session key, as the session identifier fixes all components entering the key derivation.
Ad (2). Partnered sessions agree on the contributive identifier as they contain a subset of the session identifier entries.
Ad (3). Partnered sessions agree on the intended partner as the session identifier contains both participant's identities within the `Certificate` messages.
Ad (4). More than two sessions sharing the same session or contributive identifier requires that a third session picks the same nonce and group element as one of the two sessions already partnered. The probability of such a collision can be upper-bounded by $n_s^2 \cdot 1/q \cdot 2^{-|\mathsf{nonce}|}$, where n_s is the maximum number of sessions, q is the Diffie–Hellman element group order, and $|\mathsf{nonce}| = 256$ is the bit-length of the nonces r_c and r_s.

[6] Note that the HMAC-based key derivation function HKDF in TLS 1.3 internally involves the same hash function for which we consider collision resistance breakdown. Still, we deem it reasonable to distinguish between collisions in the hash function and randomness of the HKDF output, as one property might break without the other one breaking as well. More generally, one may also instantiate HKDF based on a different hash function than the one used for computing transcript hashes.

Note that the session identifiers do not rely on any cryptographic primitive and hence the BDR-Match security bound is independent of potential Break queries issued. □

Theorem 5.2 (BDR key secrecy of TLS-(EC)DHE)**.** *The TLS 1.3 (EC)DHE handshake* TLS-(EC)DHE *achieves breakdown-resilient key secrecy for* $\mathcal{F}_{\mathsf{BDR}} = \{(\mathsf{Hash}, \mathsf{STD\text{-}Coll\text{-}Res}), (\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathsf{HMAC}, \mathsf{EUF\text{-}CMA})\}$ *with forward secrecy. More precisely, for any efficient adversary* $\mathcal{A}$ *there exist efficient adversaries* $\mathcal{B}_1$*,* ..., $\mathcal{B}_{11}$ *such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq\ & n_s^2 \cdot 2^{-|\mathsf{nonce}|} + \mathsf{Adv}^{\mathsf{COLL}}_{\mathsf{Hash},\mathcal{B}_1} + n_s \cdot \Big(n_u \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{B}_2} + n_s \cdot (\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\mathcal{B}_3} + \\ & + \mathsf{Adv}^{\mathsf{dual\text{-}PRF\text{-}sec},\mathbb{G}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_7} + \\ & + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_9} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{10}} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{HMAC},\mathcal{B}_{11}}) \Big),\end{aligned}$$

where n_s *is the maximum number of sessions,* n_u *is the maximum number of users, and* $|\mathsf{nonce}| = 256$ *is the bit-length of the nonces* r_c *and* r_s*.*

Proof. We proceed via the following sequence of games.

Game 0. The original BDR key secrecy game $G^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}}$.

Game 1. First, we exclude that two honest sessions generate the same random nonce r_c or r_s, aborting the game in such cases. The probability of this happening can be upper bounded by $n_s^2 \cdot 2^{-|\mathsf{nonce}|}$ where $|\mathsf{nonce}| = 256$ is the bit-length of the nonces r_c and r_s, i.e.,

$$\mathsf{Adv}^{G_0}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq \mathsf{Adv}^{G_1}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} + n_s^2 \cdot 2^{-|\mathsf{nonce}|}.$$

Game 2. As the next step, we exclude hash collisions (in honest sessions) prior to the breakdown of the hash function Hash. More precisely, we abort the game if in any two honest sessions' computation two distinct inputs to Hash yield the same output while $\mathsf{breakdown} = \mathsf{false}$, i.e., before $\mathcal{A}$ issued a Break query. Such a hash collision can be directly reduced to the collision resistance of Hash via a reduction $\mathcal{B}_1$ that simulates the game faithfully and aborts when the collision occurs, outputting the two input values. Hence we can bound the introduced advantage difference as

$$\mathsf{Adv}^{G_1}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq \mathsf{Adv}^{G_2}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} + \mathsf{Adv}^{\mathsf{COLL}}_{\mathsf{Hash},\mathcal{B}_1}.$$

Through this change, we are ensured that no hash collisions occur *before* the breakdown (hence, in particular, not before the test session accepts). After the breakdown, hash collisions may occur; we will see in the later game changes why those cannot affect the test session's security anymore.

Game 3. We let the challenger guess the tested session π^* and abort the game if that guess was incorrect. This can reduce the adversary's advantage by a factor of at most $\frac{1}{n_s}$, thus

$$\mathsf{Adv}^{G_2}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq n_s \cdot \mathsf{Adv}^{G_3}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}}.$$

Game 4. Next, we abort the game if the tested session receives within the `CertificateVerify` message a valid signature under the public key of some user V that no honest session of V issued. We can upper-bound the probability of such an abort by the advantage of a reduction $\mathcal{B}_2$ against the unforgeability of the signature scheme $\mathcal{S}$. Here, we use that neither can the signature scheme be broken nor can the long-term

secrets of the involved parties be corrupted before the test session has accepted. Note that forward secrecy is not affected by this game hop as a later Corrupt query on the test session's owner or partner identity does not infringe with the test session receiving an honestly generated signature at this point.

The reduction $\mathcal{B}_2$ simulates the game, guessing V and picking all but the user V's long-term keys itself. For any signature to compute for V, algorithm $\mathcal{B}_2$ queries its signing oracle. When the test session receives the forged signature, $\mathcal{B}_2$ outputs it as its own forgery. It thereby provides a sound simulation for $\mathcal{A}$ and wins in case the above abort occurs and it correctly guessed the forgery's source identity V (among the at most n_u users). Hence we can bound

$$\mathsf{Adv}^{G_3}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq \mathsf{Adv}^{G_4}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} + n_u \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{B}_2}.$$

Game 5. From now on, we are ensured that the signature obtained by the tested session π^* was honestly issued by some session π^*_{a}, which we call *associated*. Note that π^*_{a} is not necessarily partnered with π^*, but holds the same contributive identifier and is unique due to Game 1. We let the challenger guess π^*_{a} (and abort on incorrect guess), reducing the advantage of $\mathcal{A}$ by a factor at most $\frac{1}{n_s}$:

$$\mathsf{Adv}^{G_4}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq n_s \cdot \mathsf{Adv}^{G_5}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}}.$$

Game 6. The signature obtained in the test session in particular covers the (hashed) Diffie–Hellman shares sent by π^* and π^*_{a}, which the adversary hence cannot have tampered with. In particular, the adversary cannot have sent the test session Diffie–Hellman shares under a signature over a colliding hash value with some other honest session, as we excluded hash collisions prior to a breakdown in Game 2 and the test session must have accepted before any Break query is issued (as it otherwise is considered revealed).

As the next step, we can therefore replace the derived DHE value in π^* and π^*_{a} with a random group element $\widetilde{\mathrm{DHE}} \xleftarrow{\$} \mathbb{G}$. The difference in $\mathcal{A}$'s advantage introduced by this change can be bounded by the advantage of an algorithm $\mathcal{B}_3$ in breaking the DDH assumption [Bon98].[7] For this, $\mathcal{B}_3$ simulates the game truthfully, but encodes the DDH challenge values g^a, g^b in the Diffie–Hellman shares sent by π^* and π^*_{a}, and uses as value DHE in the sessions π^* and π^*_{a} the challenge value h being either g^{ab} or g^c for random c. Depending on the value h, $\mathcal{B}_3$ perfectly simulates either Game 3 or Game 4, hence establishing the bound

$$\mathsf{Adv}^{G_5}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq \mathsf{Adv}^{G_6}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} + \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\mathcal{B}_3}.$$

Game 7. At this point, $\widetilde{\mathrm{DHE}}$ in π^* and π^*_{a} is a uniformly random group element independent of all other values. This allows us to replace the handshake secret HS in both sessions with a uniformly random value $\widetilde{\mathrm{HS}} \xleftarrow{\$} \{0,1\}^\lambda$. The advantage difference introduced for $\mathcal{A}$ by this step can be bounded by a reduction $\mathcal{B}_4$ to the (dual) PRF security [Bel06, BL15] of the HKDF.Extract function when keyed with a random group element from $\mathbb{G}$ in the source key material input. For this, $\mathcal{B}_4$ relays the computation of $\mathrm{HS} \leftarrow \mathsf{HKDF.Extract}(\ldots, \widetilde{\mathrm{DHE}})$ to its PRF oracle, hence simulating either Game 4 or Game 5. Thus,

$$\mathsf{Adv}^{G_6}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq \mathsf{Adv}^{G_7}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} + \mathsf{Adv}^{\mathsf{dual\text{-}PRF\text{-}sec},\mathbb{G}}_{\mathsf{HKDF.Extract},\mathcal{B}_4}.$$

[7] Focusing on the main application traffic key tk_{app} only, which we consider derived after exchanging signatures in both directions, the DDH assumption suffices in this proof step. This is in contrast to analyses covering also the handshake traffic key (e.g., [KW16, DFGS16]) which employ the stronger pseudorandom-function oracle-Diffie–Hellman (PRF-ODH) assumption [JKSS12, BFGJ17] or the Gap-Diffie–Hellman assumption (in the random oracle model).

Games 8–13. We now replace the values $\mathrm{HTS_C}$, $\mathrm{HTS_S}$, and XHS (jointly) expanded from HS, CFK expanded from $\mathrm{HTS_C}$, MS extracted from XHS, $\mathrm{TS_S}$ and $\mathrm{TS_C}$ (jointly) expanded from MS, tk^c_{app} expanded from $\mathrm{TS_C}$ and tk^s_{app} expanded from $\mathrm{TS_S}$ in a sequence of six games with random values independently sampled from $\{0,1\}^\lambda$, in π^* and (for matching computations) π^*_a. More specifically, we replace invocations of the HKDF.Expand resp. HKDF.Extract functions in π^* and π^*_a using the respective source key by invocations of random functions. Each of these steps can be bounded in advantage difference via a reduction to the PRF security of HKDF.Expand resp. HKDF.Extract, similar to the step in Game 7.

As the PRF keys are random values chosen independently of any other value, the derived keys are independent, uniformly random values as well. This independence in particular is upheld due to the distinct PRF keys even if the adversary gains the capability to create collisions under Hash through a Break query (at some pointer after the test session accepted) and lets honest sessions compute keys under a transcript hash colliding with that of the test session, which is not excluded by Game 2.

Naming the reductions $\mathcal{B}_5, \ldots, \mathcal{B}_{10}$ we hence obtain the following bound:

$$\begin{aligned}\mathsf{Adv}^{G_7}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq & \mathsf{Adv}^{G_{13}}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\ & + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} \\ & + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_7} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} \\ & + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_9} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{10}}.\end{aligned}$$

Note that the handshake traffic keys tk^c_{hs}/tk^s_{hs} are not affected by the replacements and that, in particular, our replacements do not infringe with any honest session's capability to send and receive encrypted handshake messages under those keys.

At this point, the session key $\mathsf{K} = (tk^c_{app}, tk^s_{app})$ in the tested session π^* (and potentially π^*_a) is an independent random value. It remains to argue that the adversary cannot learn that value through a Reveal query on π^*_a.

As for the proof of Auth-NewHope (cf. Theorem 4.3), there are four possibilities for the status of the associated session π^*_a. First, π^*_a may still be running at the time of breakdown. However, as it holds the same contribute identifier as the test session (covered by the obtained signature in Game 4), this makes the adversary lose the game due to the according Finalize condition. Second, π^*_a may have rejected at the time of breakdown; in this case it does not hold a session key at all. Third, π^*_a may have accepted prior to breakdown and is partnered with π^*, hence by definition of a successful attack may not be revealed. Finally, π^*_a may have accepted prior to breakdown without being partnered with π^*, i.e., $\pi^*.\mathsf{sid} \neq \pi^*_\mathsf{a}.\mathsf{sid}$, and hence may be revealed. We will however exclude this case by showing that it implies a successful MAC forgery in the exchanged `ClientFinished` message through the following game hop.

Note that we are only interested in the case that π^*_a holds the same session key as π^*. We can therefore focus on those cases where π^*_a and π^* agree on the messages up to `ServerFinished`, as otherwise the hash value H_6 entering the session key derivation (when computing $\mathrm{TS_S}/\mathrm{TS_C}$), yielding a uniformly random key independent of that in π^*. In particular, the key derivation from the hashed transcript is not affected by a breakdown of the hash function, since π^*_a accepted prior to the breakdown.

Game 14. Let Game 14 now be as before except that the challenger aborts if π^*_a accepts with $\pi^*_\mathsf{a}.\mathsf{sid} \neq \pi^*.\mathsf{sid}$. We show that when this happens, the adversary made the server side of π^* or π^*_a accept with a forged MAC value in the `ClientFinished` message.

First of all observe that π^* and π^*_a agree on the client finished key CFK, as it is derived from DHE using the hash of `ClientHello` and `ServerHello`, all agreed upon under the shared contributive identifier by the obtained signature in Game 4. At this point, CFK was replaced in both sessions by an independent random key $\widetilde{\mathrm{CFK}}$, which enables the following reduction $\mathcal{B}_{11}$ to the EUF-CMA unforgeability of the MAC

scheme HMAC. Note that both π^* and π^*_{a} accept prior to a breakdown, hence particularly the EUF-CMA breakdown of HMAC via a Break query does not affect the argument here, as both sessions using the then exposed MAC key $\widetilde{\mathsf{CFK}}$ terminated prior to the breakdown.

In the reduction, $\mathcal{B}_{11}$ uses its MAC oracle to compute the `ClientFinished` message computed with key $\widetilde{\mathsf{CFK}}$ over $H_5 = \mathsf{Hash}(\mathtt{CH}||\ldots||\mathtt{CCV})$ exchanged between π^* and π^*_{a}. Recall that `ClientFinished` covers the (hashed) full session identifier sid, both π^* and π^*_{a} accept prior to the potential breakdown of the hash function Hash, and we excluded collisions under Hash before breakdown in Game 2. The associated session π^*_{a} accepting with a different session identifier $\pi^*_{\mathsf{a}}.\mathsf{sid} \neq \pi^*.\mathsf{sid}$ than π^* hence implies the server-side session obtained a MAC value within `ClientFinished` on a different message, hence constituting a valid existential MAC forgery.

Having $\mathcal{B}_{11}$ output the obtained `ClientFinished` MAC we can hence bound the advantage difference introduced by Game 14 as

$$\mathsf{Adv}^{G_{13}}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq \mathsf{Adv}^{G_{14}}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{HMAC},\mathcal{B}_{11}}.$$

Finally, in Game 14, the session key $\mathsf{K} = (tk^c_{app}, tk^s_{app})$ in the tested session π^* is an independent random value and the Test query thus independent of the test bit b_{test}. Furthermore, in case the associated session π^*_{a} derives the same key, the adversary is not allowed to reveal π^*_{a}. The adversary $\mathcal{A}$ hence cannot determine b_{test} better than guessing and so

$$\mathsf{Adv}^{G_{14}}_{\mathsf{TLS\text{-}(EC)DHE},\mathcal{A}} \leq 0,$$

which, together with the bounds above, completes the proof. □

5.3 Breakdown Resilience of the TLS 1.3 PSK Handshake

We now turn to the preshared-key-based TLS 1.3 handshake, focusing on the PSK-only mode. Its security only relies on the collision resistance of the hash function Hash and pseudorandomness of the key derivation function HKDF. As for the (EC)DHE handshake, we cannot hope for resilience against breakdown of the pseudorandomness of HKDF, as this may enable an adversary to distinguish real session keys from uniformly random strings. In contrast to the (EC)DHE mode, the PSK-only handshake mode in general however also does not achieve resilience against breakdown of collision resistance of the hash function Hash.

The lack of breakdown resilience for Hash is due to the deterministic key derivation from PSK using *hashed* transcripts as context values within the expansion function HKDF.Expand, where—unlike in the (EC)DHE case with its per-session DH values—potentially the *same* pre-shared key PSK is used across multiple sessions. Consider an adversary $\mathcal{A}$ that, after running an honest protocol for the test session, can find (suitable) collisions in Hash (modeled through the Break oracle). It can then run another session with an honest client, picking the server nonce in a way that the transcript of this session collides with that of the test session. To be precise, the adversary would target a collision in $H_1 = \mathsf{Hash}(\mathtt{CH}||\mathtt{SH})$ which ensures that the deterministically derived keys and MACs are the same in the test session and the colliding client's session.[8] The adversary may then reveal the colliding client's session which derives the same session key as the test session, allowing it to distinguish the Test query's output.

As a consequence, the TLS 1.3 PSK-only handshake is not breakdown resilient wrt. any of its core cryptographic components. We hence omit a (non-breakdown-resilient) security analysis and instead refer to established computational results for this mode (e.g., [DFGS15, DFGS16, BBK17]). We note that

[8] In terms of our model, the Break oracle would on input a prefix b and hash value h provide the adversary with a random preimage d such that $\mathsf{Hash}(a||b||c||d||e) = h$, where a, c, e are fixed strings (in this case matching the `CH` and `SH` message structure).

resilience against collision resistance breakdown of the hash function Hash could be achieved by using the *non-hashed* session transcript in the key derivation. However, a hashed transcript may be beneficial in terms of state and computation overhead.[9] One could furthermore argue that the attack window for a Hash breakdown may be relatively small in practice, as pre-shared keys are specified to be limited in lifetime (cf. [Res18]). Finally, when using pre-shared keys derived from the resumption master secret established in a prior full handshake, TLS 1.3 suggests that such PSKs (issued via so-called tickets) should be used only once [Res18, Sections 4.6.1 and 8.1], which, beyond privacy benefits, prevents the collision attack above.

TLS 1.3 PSK-(EC)DHE. As a final remark on TLS 1.3, we note that including Diffie–Hellman shares in the PSK-(EC)DHE handshake recovers breakdown resilience for hash collision resistance (and also achieves resilience against breakdown of the MAC scheme). Without going into further technical details, the added DHE value ensures that different sessions derive distinct session keys even under colliding hashed transcripts, following a similar argument as for the full (EC)DHE handshake (cf. Theorem 5.2). We therefore, and since it would require a security model supporting long-term pre-shared keys, omit a full analysis of the PSK-(EC)DHE here, but remark that the inclusion of Diffie–Hellman shares in the PSK-(EC)DHE handshake of TLS 1.3 hence not only achieves forward secrecy (against PSK compromise) but also breakdown resilience (for the hash and MAC function employed).

6 Strong Breakdown Resilience and Hybrid Protocols

For hybrid constructions that specifically aim to withstand cryptographic breakdowns of individual components, we demand an even stronger version of resilience: In such protocols, not only past sessions, but also ongoing and future sessions should remain secure in case a subset of the protocol's components $\mathcal{F}_{\mathsf{BDR}}$ breaks down. We term the resulting notion *strong* breakdown resilience. Note that in contrast to the regular notion of breakdown resilience from Section 3, $\mathcal{F}_{\mathsf{BDR}}$ for strong breakdown resilience will necessarily be restricted to those cryptographic components for which the (hybrid) protocol ensures some redundancy in order to maintain ongoing security.

6.1 Adjusting the Model

In order to extend the basic model from Section 3 to encompass security for future and ongoing sessions, a couple of minor changes are necessary. Figure 4 depicts the admissible Test scenarios for *strong* breakdown resilience.

Send query. The previously introduced modified $\mathsf{Send}_{\mathsf{BDR}}$ made sure that ongoing and future sessions at the time of breakdown were set to revealed and could thus not be tested by the adversary. This is no longer true for strong breakdown resilience, so we employ the original, unmodified Send query (cf. Section 2).

Contributive identifiers and state of execution at breakdown. Similarly, contributive identifiers (cid) that were needed to identify cases that are not testable (cf. Figure 1) become superfluous and any mention of them in the security definitions of BDR-Match security and BDR key secrecy (Definitions 3.1 and 3.2) are omitted. Finally, we no longer need to record the execution state at breakdown $\mathsf{st}^{\mathsf{bd}}_{\mathsf{exec}}$.

[9] This aspect is reminiscent of the comment by Dowling et al. [DFGS15] on upstream hashing in the signatures sent in the TLS 1.3 (EC)DHE handshake.

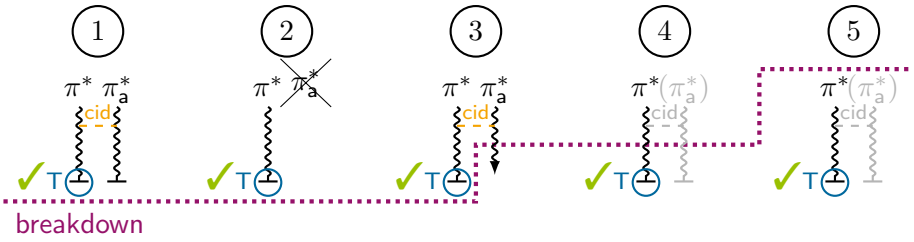


Figure 4: Illustration of permissible Test queries for strong breakdown resilience. Scenarios 1 and 2 are as in Figure 1; Scenarios 3, 4, and 5 are now permissible.

Break oracle. To model the Break oracle for hybrid protocols we introduce a $(\mathsf{KE}, \mathsf{BR})$ entry for breaking BR key secrecy (cf. Table 1). When calling the Break oracle, all the session keys of accepted sessions are disclosed to the adversary and the oracle proceeds as specified before in the model.

6.2 Hybrid Key Exchange Security

We can now leverage our strong breakdown resilience model to investigate the security of hybrid key exchange protocols. These usually combine two (or more) key agreement components such that the overall protocol remains secure (for past *and* future sessions) if at least one of the component schemes remains secure. The idea of combining two cryptographic schemes of the same type for robustness is not new and has been studied extensively in the literature, often referred to as *combiners* (e.g., [EG85, ZHSI04, DK05, HKN+05]).

Recently, hybrid key exchanges, combining classically secure and quantum-resistant schemes, have gained a lot of interest. Such hybrids offer a way to transition to post-quantum solutions with security against (future) quantum adversaries, while still maintaining the security guarantees that are offered today by well-established classical key agreement. The latter guarantee is mandated by the uncertainty about which post-quantum hardness assumptions to rely on and how to select appropriately strong parameter. In 2018, Giacon et al. [GHP18] initiated the study of KEM combiners and with it, implicitly, hybrid (unauthenticated) key exchange. Bindel et. al. [BBF+19] extended this line of work by proposing a model for hybrid authenticated key exchange with respect to different levels of quantum adversaries, focusing however exclusively on combiners of classical and post-quantum KEMs.

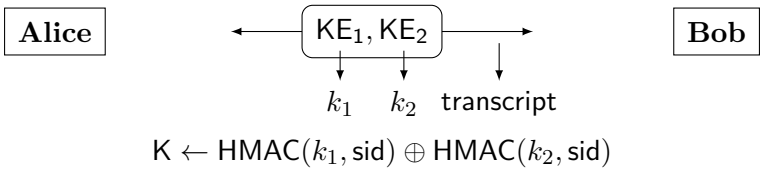


Figure 5: Hybrid protocol Π from authenticated key exchange protocols KE_1 and KE_2.

Going beyond KEM-based combiners, we in the following illustrate how our notion of strong breakdown resilience can capture the combiner-type security of an arbitrary hybrid key exchange construction. To this end, consider the simple and generic protocol Π depicted in Figure 5 which combines two arbitrary, independently secure authenticated key exchange protocols KE_1 and KE_2 (yielding shared secret keys k_1, resp. k_2). The run of the two protocol is followed by some joint post-processing via $\mathsf{HMAC}(k_1, \mathsf{sid}) \oplus \mathsf{HMAC}(k_2, \mathsf{sid})$. Note that each key is applied to the full session identifier $\mathsf{sid} = (\mathsf{sid}_1, \mathsf{sid}_2)$ of both executions.[10]

[10]Instead of using the exclusive-or one could also rely on the dual pseudorandomness of HMAC and instead compute the

For technical reasons, rooted in the security model with a single Test query, we also need that the session identifiers $\mathsf{sid}_1, \mathsf{sid}_2$ in each of the key exchange protocols are derived efficiently from the transcript. We call such session identifiers *public*. Our choice for sid also straightforwardly ensures Match security via the security of both underlying protocols:

1. Identical session identifiers $\mathsf{sid} = \mathsf{sid}' \neq \bot$ imply identical identifiers $\mathsf{sid}_1 = \mathsf{sid}'_1$ and $\mathsf{sid}_2 = \mathsf{sid}'_2$. This, in turn, means via the Match security of KE_1 and KE_2 identical keys $k_1 = k'_1$ and $k_2 = k'_2$, and thus that also the combined session keys match.

2. Any mismatch concerning the intended partner (for identical session identifiers $\mathsf{sid} = \mathsf{sid}'$) or more than two colliding session identifiers $\mathsf{sid} = \mathsf{sid}' = \mathsf{sid}''$ immediately yield contradictions for the underlying protocols.

We stress that we need Match security of both protocols in order to argue security for the combined protocol. Fortunately, Match security of a protocol usually relies on statistical properties like collision-intractability of nonces, such that breaks of cryptographic assumptions are irrelevant. We may therefore indeed assume that both protocol have this property simultaneously.

We can then formally show—through Theorem 6.1 below—that Π achieves strong breakdown resilience, assuming HMAC is a secure pseudorandom function, and under security breakdowns of either KE_1 or KE_2: independent of whether an adversary at some point obtains breaking capabilities for (either) key exchange protocol KE_1 or KE_2, its success (probability) in breaking protocol Π is upper bounded by the maximum of both advantages of regular adversaries in breaking KE_1 and KE_2 separately (plus the security advantage against the pseudorandomness of HMAC). This is so, intuitively, as secrecy of either key implies that the HMAC computation yields a pseudorandom value.

Theorem 6.1 (Strong BDR Key Secrecy of generic hybrid KE)**.** *Let* KE_1 *and* KE_2 *be* BR*-secure key exchange protocols with public session identifiers. Let* HMAC *be a pseudorandom function.*

Then the protocol $\Pi = [\mathsf{KE}_1, \mathsf{KE}_2, \mathsf{HMAC}]$ *given in Figure 5 achieves strong breakdown-resilient key secrecy for* $\mathcal{F}_{\mathsf{BDR}} \in \{\{(\mathsf{KE}_1, \mathsf{BR})\}, \{(\mathsf{KE}_2, \mathsf{BR})\}\}$*. More precisely, for any efficient adversary* $\mathcal{A}$*, there exist efficient adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$ *such that*

$$\mathsf{Adv}^{\mathsf{sBDR}(\{\mathsf{KE}_\beta,\mathsf{BR}\}),\mathcal{D}}_{\Pi,\mathcal{A}} \leq \; n_s \cdot (\mathsf{Adv}^{\mathsf{BR}}_{\mathsf{KE}_{3-\beta},\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HMAC},\mathcal{B}_2}),$$

where n_s *is the maximum number of sessions.*

Proof. Assume for simplicity below that KE_1 is secure and KE_2 is susceptible to a breakdown. The other case follows by symmetry.

Game 0. The original sBDR key secrecy game $G^{\mathsf{sBDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}_{\Pi,\mathcal{A}}$.

Game 1. We start by guessing the first accepting session which will hold the same session identifier sid_1 (of the KE_1 part) as the test session. Denote this first session as π^*_{a}. Note that this could be the test session itself. The guessing strategy reduces $\mathcal{A}$'s advantage by a factor of at most $\frac{1}{n_s}$, i.e., $\mathsf{Adv}^{G_0}_{\Pi,\mathcal{A}} \leq n_s \cdot \mathsf{Adv}^{G_1}_{\Pi,\mathcal{A}}$.

Game 2. Next replace all derived key parts k_1 in (subsequently accepting) sessions with the same identifier sid_1 as π^*_{a} consistently by the same random key $\tilde{k}_1$. Note that since session identifiers of KE_1 are public, such sessions are easy to determine given the transcripts.

key as $\mathsf{HMAC}(\mathsf{HMAC}(k_1, \mathsf{sid}), \mathsf{HMAC}(k_2, \mathsf{sid}))$ or $\mathsf{HMAC}(\mathsf{HMAC}(k_1, k_2), \mathsf{sid})$, resembling TLS key derivation more closely. We use the simpler version here which does not require an additional assumption of HMAC beyond pseudorandomness.

Any difference in the advantages of $\mathcal{A}$ between Games 1 and 2 can be bounded by an adversary $\mathcal{B}_1$ against the BR key secrecy of KE_1. The reduction would simulate all KE_2 steps internally, such that it knows all key parts k_2 of any session. In particular, it can also answer $\mathcal{A}$'s breakdown queries by supplying all session key parts for KE_2. Adversary $\mathcal{B}_1$ calls its Test oracle (for KE_1) about session π_{a}^* to get either k_1 or a random key $\tilde{k}_1$. It subsequently uses this value to compute the session keys of the combiner according to the protocol in all sessions with the same identifier sid_1. For all other sessions, also preceding ones, it calls its Reveal oracle to get the k_1 key part and uses the returned value to compute the session keys of the combined protocol. In addition, for $\mathcal{A}$'s query Test (to the combiner), algorithm $\mathcal{B}_1$ chooses a random bit and returns either the computed session key or a random key.

For the analysis note that, by assumption, π_{a}^* is the first completed session with sid_1. Hence, no Reveal query of $\mathcal{B}_1$ can violate freshness of the Test session of $\mathcal{B}_1$. It follows that the simulation is sound and perfectly mimics the difference between the two games, depending on whether $\mathcal{B}_1$'s test bit is 0 or 1. Hence, the advantage of $\mathcal{A}$ is upper-bounded by the advantage of $\mathcal{B}_1$ against KE_1: $\mathsf{Adv}^{G_1}_{\Pi,\mathcal{A}} \leq \mathsf{Adv}^{G_2}_{\Pi,\mathcal{A}} + \mathsf{Adv}^{\mathsf{BR}}_{\mathsf{KE}_1,\mathcal{B}_1}$.

Game 3. Next, replace $\mathsf{HMAC}(\tilde{k}_1, \mathsf{sid})$ in all sessions with the session identifier part sid_1 as π_{a}^* by an independent random value $\tilde{h}_{\mathsf{sid}}$ (but consistently for all identical full identifiers sid).

The advantage of an adversary distinguishing this game from the previous one is bounded by the advantage of an adversary $\mathcal{B}_2$ breaking the PRF security of HMAC. The reduction is straightforward, simulating all steps of the key exchange game and calling an external random or pseudorandom function oracle. Thus, $\mathsf{Adv}^{G_2}_{\Pi,\mathcal{A}} \leq \mathsf{Adv}^{G_3}_{\Pi,\mathcal{A}} + \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{HMAC},\mathcal{B}_2}$.

At this point, the session key of $\mathcal{A}$'s tested session is always distributed uniformly, independently of the secret test bit b_{test}. This is so since the key is either random or it is derived as the exclusive-or with a random string. The only difference between the two cases is that, if the test oracle returns the actual key ($b_{\mathsf{test}} = 1$), then any partnered session with the same value sid holds the same key. In contrast, if the returned key is random ($b_{\mathsf{test}} = 0$), then such session partners would most likely hold a different key. However, since the adversary $\mathcal{A}$ cannot Reveal the key of a session with the same overall session identifier as the test session without losing the game, the two cases are identical concerning $\mathcal{A}$'s advantage, i.e., $\mathsf{Adv}^{G_3}_{\Pi,\mathcal{A}} \leq 0$. □

7 Conclusion

We presented the first extension to a variant of the widely used Bellare–Rogaway model [BR94] for authenticated key exchange which allows to assess the impact of a break of cryptographic building blocks on already completed sessions. The resulting security notion is termed *breakdown resilience*. We showed that both an authenticated version of NewHope as well as the TLS 1.3 (EC)DHE handshake mode achieve breakdown resilience for varying broken primitives. The case of the TLS 1.3 PSK(-only) mode illustrates that seemingly minor design choices can significantly impact the breakdown resilience of protocols. We furthermore showed how a modified version of the breakdown resilience model can be used to argue about the security of hybrid key exchange constructions.

We are confident that the presented ideas can also be integrated into other relevant models for authenticated key exchange, such as the CK model [CK01], its extension eCK [LLM07], the ACCE model [JKSS12], as well as the multi-stage setting [FG14]. Moreover, the notion may even be transferred to different classes of cryptographic protocols.

Acknowledgments

Felix Günther is supported in part by Research Fellowship grant GU 1859/1-1 of the DFG and National Science Foundation (NSF) grants CNS-1526801 and CNS-1717640. This work has been co-funded by the DFG as part of project S4 within the CRC 1119 CROSSING and as part of project D.2 within the RTG 2050 "Privacy and Trust for Mobile Users."

References

[AAB+19] Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Thomas Pöppelmann, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope: Algorithm specifications and supporting documentation. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/NewHope-Round2.zip, March 2019. Accessed: 2019-04-24. (Cited on pages 4, 5, and 18.)

[ABD+15] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 5–17, Denver, CO, USA, October 12–16, 2015. ACM Press. (Cited on page 3.)

[ABP+13] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. On the security of RC4 in TLS. In Samuel T. King, editor, *USENIX Security 2013: 22nd USENIX Security Symposium*, pages 305–320, Washington, DC, USA, August 14–16, 2013. USENIX Association. (Cited on page 3.)

[ADPS16a] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. NewHope without reconciliation. Cryptology ePrint Archive, Report 2016/1157, 2016. http://eprint.iacr.org/2016/1157. (Cited on page 18.)

[ADPS16b] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 327–343, Austin, TX, USA, August 10–12, 2016. USENIX Association. (Cited on pages 3, 7, 13, and 18.)

[BBF+16] 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. (Cited on page 7.)

[BBF+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In *Post-Quantum Cryptography*, Cham, 2019. Springer International Publishing. (Cited on pages 3, 7, and 33.)

[BBK17] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *2017 IEEE Symposium on Security and Privacy*, pages 483–502, San Jose, CA, USA, May 22–26, 2017. IEEE Computer Society Press. (Cited on pages 7, 24, and 31.)

[BCD+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1006–1018, Vienna, Austria, October 24–28, 2016. ACM Press. (Cited on page 3.)

[BCGP08] Colin Boyd, Yvonne Cliff, Juan González Nieto, and Kenneth G. Paterson. Efficient one-round key exchange in the standard model. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 08: 13th Australasian Conference on Information Security and Privacy*, volume 5107 of *Lecture Notes in Computer Science*, pages 69–83, Wollongong, Australia, July 7–9, 2008. Springer, Heidelberg, Germany. (Cited on pages 6 and 15.)

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany. (Cited on page 25.)

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press. (Cited on page 3.)

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *IEEE EuroS&P 18*, pages 353–367. IEEE Computer Society Press, April 2018. (Cited on pages 3, 6, and 15.)

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany. (Cited on page 29.)

[BFG19] Jacqueline Brendel, Marc Fischlin, and Felix Günther. Breakdown resilience of key exchange protocols: NewHope, TLS 1.3, and hybrids. Cryptology ePrint Archive, Report 2017/1252, 2019. https://eprint.iacr.org/2017/1252. (Cited on page 13.)

[BFGJ17] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 651–681, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. (Cited on page 29.)

[BFK+14] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Santiago Zanella Béguelin. Proving the TLS handshake secure (as it is). In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 235–255, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. (Cited on page 12.)

[BFWW11] 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. (Cited on page 9.)

[BL15] Mihir Bellare and Anna Lysyanskaya. Symmetric and dual PRFs from standard assumptions: A generic validation of an HMAC assumption. Cryptology ePrint Archive, Report 2015/1198, 2015. http://eprint.iacr.org/2015/1198. (Cited on page 29.)

[BL16] Karthikeyan Bhargavan and Gaëtan Leurent. Transcript collision attacks: Breaking authentication in TLS, IKE and SSH. In *ISOC Network and Distributed System Security Symposium – NDSS 2016*, San Diego, CA, USA, February 21–24, 2016. The Internet Society. (Cited on page 3.)

[Bon98] Dan Boneh. The decision diffie-hellman problem. In Joe P. Buhler, editor, *Algorithmic Number Theory: Third International Symposiun*, pages 48–63, Berlin, Heidelberg, 1998. Springer. (Cited on page 29.)

[BR94] 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, Heidelberg, Germany. (Cited on pages 4, 7, and 35.)

[Bra16] Matt Braithwaite. Google Security Blog: Experimenting with post-quantum cryptography. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html, July 2016. Accessed: 2019-04-24. (Cited on pages 3 and 18.)

[Brz13] Christina Brzuska. *On the Foundations of Key Exchange*. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2013. http://tuprints.ulb.tu-darmstadt.de/3414/. (Cited on page 9.)

[CCG16] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On Post-compromise Security. In *IEEE CSF 16*, pages 164–178, 2016. (Cited on pages 3 and 6.)

[CF12] 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, Heidelberg, Germany. (Cited on page 12.)

[CHH+17] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1773–1788, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. (Cited on pages 7 and 24.)

[CHSvdM16] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication. In *2016 IEEE Symposium on Security and Privacy*, pages 470–485, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press. (Cited on pages 7 and 24.)

[CK01] 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, Heidelberg, Germany. (Cited on pages 3, 6, and 35.)

[dB94] Bert den Boer and Antoon Bosselaers. Collisions for the compressin function of MD5. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 293–304, Lofthus, Norway, May 23–27, 1994. Springer, Heidelberg, Germany. (Cited on page 3.)

[DFGS15] 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. (Cited on pages 7, 12, 24, 25, 31, and 32.)

[DFGS16] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 draft-10 full and pre-shared key handshake protocol. Cryptology ePrint Archive, Report 2016/081, 2016. http://eprint.iacr.org/2016/081. (Cited on pages 7, 24, 25, 29, and 31.)

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany. (Cited on page 33.)

[DS15] Benjamin Dowling and Douglas Stebila. Modelling ciphersuite and version negotiation in the TLS protocol. In Ernest Foo and Douglas Stebila, editors, *ACISP 15: 20th Australasian Conference on Information Security and Privacy*, volume 9144 of *Lecture Notes in Computer Science*, pages 270–288, Brisbane, QLD, Australia, June 29 – July 1, 2015. Springer, Heidelberg, Germany. (Cited on page 7.)

[DVOW92] Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992. (Cited on pages 3 and 6.)

[EG85] S. Even and Oded Goldreich. On the power of cascade ciphers. *ACM Trans. Comput. Syst.*, 3(2):108–116, 1985. (Cited on page 33.)

[FG14] 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. (Cited on pages 25 and 35.)

[FG17] Marc Fischlin and Felix Günther. Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates. In *IEEE EuroS&P 17*, pages 60–75. IEEE Computer Society Press, April 2017. (Cited on pages 7 and 24.)

[FGSW16] 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. (Cited on pages 7 and 24.)

[GCR16] Ilias Giechaskiel, Cas J. F. Cremers, and Kasper Bonne Rasmussen. On bitcoin security in the presence of broken cryptographic primitives. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016: 21st European Symposium on Research in Computer Security, Part II*, volume 9879 of *Lecture Notes in Computer Science*, pages 201–222, Heraklion, Greece, September 26–30, 2016. Springer, Heidelberg, Germany. (Cited on page 6.)

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. (Cited on pages 3 and 33.)

[GMPS14] Sourav Sen Gupta, Subhamoy Maitra, Goutam Paul, and Santanu Sarkar. (Non-)random sequences from (non-)random permutations - analysis of RC4 stream cipher. *Journal of Cryptology*, 27(1):67–108, January 2014. (Cited on page 3.)

[Gün90] Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37, Houthalen, Belgium, April 10–13, 1990. Springer, Heidelberg, Germany. (Cited on pages 3 and 6.)

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. (Cited on page 33.)

[JKSS12] 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, Heidelberg, Germany. (Cited on pages 29 and 35.)

[Kra03] Hugo Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. (Cited on page 5.)

[Kra10] 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, Heidelberg, Germany. (Cited on page 25.)

[Kra16] Hugo Krawczyk. A unilateral-to-mutual authentication compiler for key exchange (with applications to client authentication in TLS 1.3). In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1438–1450, Vienna, Austria, October 24–28, 2016. ACM Press. (Cited on pages 7 and 24.)

[KW16] Hugo Krawczyk and Hoeteck Wee. The OPTLS protocol and TLS 1.3. In *IEEE EuroS&P 16*, pages 81–96. IEEE Computer Society Press, March 2016. (Cited on page 29.)

[Lan18] Adam Langley. Imperial Violet: Cecpq2. https://www.imperialviolet.org/2018/12/12/cecpq2.html, December 2018. Accessed: 2019-04-24. (Cited on page 3.)

[LLM07] 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, Heidelberg, Germany. (Cited on pages 6 and 35.)

[LXZ+16] Xinyu Li, Jing Xu, Zhenfeng Zhang, Dengguo Feng, and Honggang Hu. Multiple handshakes security of TLS 1.3 candidates. In *2016 IEEE Symposium on Security and Privacy*, pages 486–505, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press. (Cited on pages 7 and 24.)

[NIS17] NIST. Post-quantum cryptography standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization, Jan 2017. Accessed: 2019-04-24. (Cited on pages 3 and 18.)

[PvdM16] Kenneth G. Paterson and Thyla van der Merwe. Reactive and proactive standardisation of TLS. In *Security Standardisation Research (SSR 2016)*, pages 160–186, 2016. (Cited on pages 7 and 24.)

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. (Cited on pages 4, 5, 13, 24, 26, and 32.)

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 570–596, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. (Cited on page 3.)

[SKP16] Marc Stevens, Pierre Karpman, and Thomas Peyrin. Freestart collision for full SHA-1. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 459–483, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. (Cited on page 3.)

[SLdW07] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany. (Cited on page 3.)

[Ste13] Marc Stevens. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 245–261, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. (Cited on page 3.)

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. (Cited on page 3.)

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. (Cited on page 3.)

[ZHSI04] Rui Zhang, Goichiro Hanaoka, Junji Shikata, and Hideki Imai. On the security of multiple encryption or CCA-security+CCA-security=CCA-security? In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 360–374, Singapore, March 1–4, 2004. Springer, Heidelberg, Germany. (Cited on page 33.)

A Security Assumptions

Definition A.1 ((Public Key) IND-CCA2 Security)**.** *Let λ be the security parameter. Furthermore let $\mathcal{E} = (\mathsf{KG}, \mathit{Enc}, \mathit{Dec})$ be a public key encryption scheme and let $\mathcal{A}$ be a PPT algorithm. We define the following* IND-CCA2 *security game $G^{\text{IND-CCA2}}_{\mathit{Enc},\mathcal{A}}(\lambda)$:*

Setup. *Generate a key pair $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$ and give pk to the adversary $\mathcal{A}$.*

Query Phase 1. *In the next phase $\mathcal{A}$ can adaptively query polynomially many messages m to the encryption oracle and polynomially many ciphertexts to the decryption oracle.*

Challenge Phase. *The adversary $\mathcal{A}$ submits two distinct messages m_0, m_1 to the challenger. The challenger chooses a bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random, and returns the challenge ciphertext $c^* = \mathit{Enc}(pk, m_b)$ to the adversary.*

Query Phase 2. *The adversary may make further (polynomially many) calls to the encryption and decryption oracle with the sole limitation that $\mathcal{A}$ may not query the challenge ciphertext c^* to the decryption oracle.*

Output. *At some point, $\mathcal{A}$ outputs a bit b'. Output 1 iff $b = b'$.*

We define the advantage function as

$$\mathsf{Adv}^{\text{IND-CCA2}}_{\mathcal{E},\mathcal{A}}(\lambda) := \Pr\left[G^{\text{IND-CCA2}}_{\mathit{Enc},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}.$$

We say that a public key encryption scheme $\mathcal{E}$ is IND-CCA2 secure*, if for any PPT adversary $\mathcal{A}$ the advantage function is negligible (as a function in λ).*

Definition A.2 (KEM IND-CPA Security)**.** *Let λ be the security parameter. Furthermore let $\mathsf{KEM} = (\mathsf{KG}, \mathsf{Encaps}, \mathsf{Decaps})$ be a key encapsulation mechanism and let $\mathcal{A}$ be a PPT algorithm. We define the following* IND-CPA *security game $G^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$:*

Setup and Challenge *Generate a key pair $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$. The challenger computes $c^*, k_0^* \xleftarrow{\$} \mathsf{Encaps}(pk)$ and chooses k_1^* at random from the key space. The challenger flips a bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random, and gives pk, c^* and k_b^* to the adversary.*

Output. *At some point, $\mathcal{A}$ outputs a bit b'. Output 1 iff $b = b'$.*

We define the advantage function as

$$\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := \Pr\left[G^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}.$$

We say that a key encapsulation mechanism KEM *is* IND-CPA secure*, if for any PPT adversary* $\mathcal{A}$ *the advantage function is negligible (as a function in* λ*).*

Definition A.3 (EUF-CMA Security)**.** *Let* λ *be the security parameter. Furthermore let* $\mathcal{S} = (\mathsf{SKG}, \mathsf{Sig}, \mathsf{SVf})$ *be a signature scheme and let* $\mathcal{A}$ *be a PPT algorithm. We define the following* EUF-CMA *security game* $G^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(\lambda)$*:*

Setup. *Generate a key pair* $(pk, sk) \xleftarrow{\$} \mathsf{SKG}(1^\lambda)$ *and give* pk *to the adversary* $\mathcal{A}$*.*

Query Phase. *In the next phase* $\mathcal{A}$ *can adaptively query messages* $m_1, m_2, \ldots, m_q \in \{0,1\}^*$ *with* $q \in \mathbb{N}$ *arbitrary, which the signing oracle answers with* $\sigma_1 \leftarrow \mathsf{Sig}(sk, m_1), \sigma_2 \leftarrow \mathsf{Sig}(sk, m_2), \ldots, \sigma_q \leftarrow \mathsf{Sig}(sk, m_q)$*.*

Output. *At some point,* $\mathcal{A}$ *outputs a message* m^* *and a potential signature* σ^**. Output* 1 *iff* $\mathsf{SVf}(pk, m^*, \sigma^*) = 1$ *and* $m^* \neq m_i$ *for all* $i = 1, 2, \ldots, q$*.*

We define the advantage function as

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(\lambda) := \Pr\left[G^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(\lambda) = 1\right].$$

We say that a signature scheme $\mathcal{S}$ *is* EUF-CMA secure*, if for any PPT adversary* $\mathcal{A}$ *the advantage function is negligible (as a function in* λ*).*

The unforgeability of a message authentication scheme $\mathcal{M} = (\mathsf{MKG}, \mathsf{MAC}, \mathsf{MVf})$ is defined analogously.

Definition A.4 (Collision Resistance)**.** *Let* λ *be the security parameter. Furthermore let* $\mathcal{H} = (\mathsf{HKG}, \mathsf{Hash})$ *be a hash function and let* $\mathcal{A}$ *be a PPT algorithm. We define the following* Coll-Res *security game* $G^{\mathsf{Coll\text{-}Res}}_{\mathcal{H},\mathcal{A}}(\lambda)$*:*

Setup. *Generate a key* $s \xleftarrow{\$} \mathsf{HKG}(1^\lambda)$ *and give* s *to the adversary* $\mathcal{A}$*.*

Output. *At some point,* $\mathcal{A}$ *outputs* x, x'*. Output* 1 *iff* $x \neq x'$ *and* $\mathsf{Hash}^s(x) = \mathsf{Hash}^s(x')$*.*

We define the advantage function as

$$\mathsf{Adv}^{\mathsf{Coll\text{-}Res}}_{\mathcal{H},\mathcal{A}}(\lambda) := \Pr\left[G^{\mathsf{Coll\text{-}Res}}_{\mathcal{H},\mathcal{A}}(\lambda) = 1\right].$$

We say that a hash function $\mathcal{H}$ *is* collision resistant*, if for any PPT adversary* $\mathcal{A}$ *the advantage function is negligible (as a function in* λ*).*

Definition A.5 (PRF Security)**.** *Let* λ *be the security parameter. Furthermore let* $F : K \times X \to Y$ *be a* PRF *and let* $\mathcal{A}$ *be a PPT algorithm. We define the following* PRF-sec *security game* $G^{\mathsf{PRF\text{-}sec}}_{F,\mathcal{A}}(\lambda)$*:*

Setup. *Sample a bit* $b \xleftarrow{\$} \{0,1\}$*, a key* $k \xleftarrow{\$} K$*, and some* $f \xleftarrow{\$} \mathsf{Fun}[X \to Y]$*, where* $\mathsf{Fun}[X \to Y]$ *denotes the set of all functions from* X *to* Y*.*

Query Phase *The adversary* $\mathcal{A}$ *may now query polynomially many labels* $x_i \in X$ *to the challenger and, depending on the bit* b*, receives either the value* $F(k, x_i)$ *for* $b = 0$ *or* $f(x_i)$ *for* $b = 1$*.*

Output. *At some point, $\mathcal{A}$ outputs its guess b'. Output 1 iff $b = b'$.*

We define the advantage function as

$$\mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{F,\mathcal{A}}(\lambda) := \Pr\left[G^{\mathsf{PRF\text{-}sec}}_{F,\mathcal{A}}(\lambda) = 1\right].$$

We say that F is PRF-secure*, if for any PPT adversary $\mathcal{A}$ the advantage function is negligible (as a function in λ).*

Definition A.6 (KDF Security)**.** *Let λ be the security parameter. Let $\mathsf{kdf} : \Sigma \times \mathbb{N} \times \mathsf{Salt} \times \mathsf{Context} \to \{0,1\}^l$ be a key derivation function with inputs source keying material σ from Σ, $l \in \mathbb{N}$ the output length and optional parameters $s \in \mathsf{Salt}$ and $c \in \mathsf{Context}$. Furthermore, let $\mathcal{A}$ be a PPT algorithm. We define the following* KDF-sec *security game $G^{\mathsf{KDF\text{-}sec}}_{\mathsf{kdf},\mathcal{A}}(\lambda)$:*

Setup. *Sample (secret) keying material σ with auxiliary information a from source Σ, as well as a salt value $s \xleftarrow{\$} \mathsf{Salt}$ from all possible salt values. Give s and a to the adversary $\mathcal{A}$.*

Query Phase 1. *The adversary $\mathcal{A}$ may now query polynomially many pairs $(l_i, c_i) \in \mathbb{N} \times \mathsf{Context}$ to the challenger and receives the values $\mathsf{kdf}(\sigma, l_i, s, c_i)$.*

Challenge Phase. *The adversary $\mathcal{A}$ submits (l^*, c^*) to the challenger. The challenger chooses a bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random, and depending on the bit b returns the challenge y_b where $y_0 = \mathsf{kdf}(\sigma, l^*, s, c^*)$ for $b = 0$ and $y_1 \xleftarrow{\$} \{0,1\}^{l^*}$ for $b = 1$ to the adversary.*

Query Phase 2. *The adversary may make (polynomially many) further calls to the kdf oracle as in Query Phase 1, with the sole limitation that $\mathcal{A}$ may not query the challenge pair (l^*, c^*).*

Output. *At some point, $\mathcal{A}$ outputs its guess b'. Output 1 iff $b = b'$.*

We define the advantage function as

$$\mathsf{Adv}^{\mathsf{KDF\text{-}sec}}_{\mathsf{kdf},\mathcal{A}}(\lambda) := \Pr\left[G^{\mathsf{KDF\text{-}sec}}_{\mathsf{kdf},\mathcal{A}}(\lambda) = 1\right].$$

We say that kdf is KDF-secure*, if for any PPT adversary $\mathcal{A}$ the advantage function is negligible (as a function in λ).*