Breakdown Resilience of Key Exchange Protocols and the Cases of NewHope and TLS 1.3

Jacqueline Brendel, Marc Fischlin, and Felix Günther

Cryptoplexity, Technische Universität Darmstadt, Germany
{jacqueline.brendel, marc.fischlin, felix.guenther}@cryptoplexity.de

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. Remarkably, the authentication step in such protocols is usually still carried out with quantum-vulnerable signature schemes. The intuition here is that the adversary would need to break this protocol primitive today, without having quantum power yet. The question we address here is if this intuition is justified, and if so, if we can show this rigorously.
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 sessions. In this work we introduce an extension to the common Bellare–Rogaway model that can provide security guarantees in what we call the breakdown scenario and we term the resulting security notion *breakdown resilience*. The model allows to make security claims even in case of unexpected failure of primitives in the protocol, may it be hash functions, signature schemes, key derivation functions, etc. To validate the proposed security model with respect to real-world protocols we show that breakdown resilience for certain primitives is achieved by both an authenticated variant of the recently introduced post-quantum secure key exchange protocol NewHope (Alkim et al., USENIX Security 2016), as well as by TLS 1.3, which is currently being developed by the Internet Engineering Task Force.

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 guarantee of the protocol is 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.

Particularly in the area of key exchange, protocols 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 indeed one expects future sessions to be 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? The notion of forward secrecy answers this question only partially, as we will see, for the usage of long-term secrets. An holistic notion of security against later breakdowns of arbitrary (keyed and unkeyed) primitives as well as cryptographic hardness assumptions is lacking, though.

1.1 Breakdowns and Mitigations in Practical Key Exchange Protocols

The lack 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 [36,2] over poor Diffie–Hellman parameter choices [1] to advances in breaking widely deployed hash functions like MD5 [26,55,53] or SHA-1 [54,50,52,51] also enabling key-exchange-level attacks [11].

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., [15,3,14]) and in parts experimentally deployed today (e.g., [17]). 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, the authors of the quantum-secure key-exchange protocol NewHope argue that "[...] attacks on the [classical] signature will not compromise previous communication" [3]. While this intuition may be correct, there are no formal justifications for such statements at this point.

1.2 Our Contributions

There is hence a need for a formal tool to assess the precise security of key exchange protocols in case (some) underlying primitives or hardness assumptions break. To this end, 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. [3] as well as draft-21 of the upcoming Transport

Layer Security protocol version 1.3 [48] developed by the Internet Engineering Task Force (IETF) with respect to their resilience against such breakdowns.

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 [7] (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.

The resulting novel 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.

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

[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 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.

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, a post-quantum key exchange protocol proposed by Alkim et al. [3] based on previous work by Bos et al. [15]. The protocol gained widespread attention and experimental deployment in Google Chrome Canary [17]. For this, we first define AUTH-NEWHOPE as a natural, authenticated version of NEWHOPE as envisioned by the authors of the latter by employing authentication via (classical) signatures and MACs following the SIGMA (SIGn-and-MAC) approach proposed by Krawczyk [39], which has been adopted in major Internet security protocols like IPsec and TLS.

Using our novel 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 well as for the extendable-output function employed in the protocol modeled as a random oracle. As the AUTH-NEWHOPE protocol employs a generic SIGMA-style [39] authentication step following the basic NEWHOPE protocol, 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 exchange 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) specified in the draft-21 of the upcoming Transport Layer Security protocol, TLS 1.3 [48]. 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, (strong) hash collision attacks 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).

1.3 Related Work and Delineation

Our work shares or extends and definitely is inspired by conceptual ideas of prior work on the security of both key exchange specifically and cryptographic protocols more broadly. Yet, the notion of breakdown resilience we introduce 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 [35,27,21]. 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 [21,44]) in that our focus is on the breakdown of complete primitives or hardness assumptions rather than on the exposure of specific protocol values. Furthermore, the breakdown resilience scenario 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 [16] like the recently proposed scheme based on the Kyber KEM [13]. 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.

Bitcoin security in the presence of broken primitives. Giechaskiel, Cremers, and Rasmussen [34] were the first to systematically explore how broken or weakened hash functions and/or signatures affect the security of Bitcoin. While their study was 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 in spirit to other kinds of protocols as well.

Post-compromise security. With their notion of *post-compromise security*, Cohn-Gordon, Cremers, and Garratt [22] establish security guarantees for communication *after* participants have been compromised to various degrees. 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. Our notion is furthermore concerned with the security of sessions that were completed *before* such a breakdown occurred.

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. [9] and Dowling and Stebila [30], 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 [3]. We augment NewHope to include authentication and study its security not only as an authenticated key exchange 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 [28,29,24,33,45,41,32,8,23] and also refer to [47] 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.

2 The Bellare–Rogaway Model for Authenticated Key Exchange

We begin by recapping key exchange security in the style of the model by Bellare and Rogaway [7] 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 $\mathsf{Corrupt}$ 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. The default value upon creation of the session is $\mathsf{running}$.
- $\mathsf{sid} \in \{0,1\}^* \cup \{\bot\}$ indicates the session identifier. The default value is $\bot$.
- $\mathsf{st}_{\mathsf{key}} \in \{\mathsf{fresh}, \mathsf{revealed}\}$ indicates the state of the session key K. The default value is fresh.
- $\mathsf{K} \in \{0,1\}^* \cup \{\bot\}$ indicates the established session key. The default value is $\bot$.
- $\mathsf{tested} \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether the session key K has been tested or not. Default value for each key is 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., $\pi^k_{U,V}.\mathsf{sid} = \pi^{k'}_{V',U'}.\mathsf{sid} \neq \bot$. We require that any non-tampered 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 the 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. [20,19] to separate the overall BR security properties into the notions of $\mathsf{BR\text{-}Match}$ security and BR key secrecy. The conditions of $\mathsf{BR\text{-}Match}$ security guarantee that the session identifiers sid ensure an appropriate identification of partnered sessions. BR key secrecy then ensures a protocol establishes session keys that are indistinguishable from random strings and (implicitly) mutually authenticated.

Definition 1 ($\mathsf{BR\text{-}Match}$ Security).
Let λ be the security parameter. Furthermore let KE be 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 $\pi.\mathsf{sid} = \pi'.\mathsf{sid} \neq \bot$, 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 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.)*
3. *There exist at least three sessions π, π', and π'' such that π, π', π'' are pairwise distinct, but $\pi.\mathsf{sid} = \pi'.\mathsf{sid}' = \pi''.\mathsf{sid} \neq \bot$. (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 (BR **Key Secrecy**)**.** *Let λ be the security parameter. Furthermore let* KE *be a key exchange protocol with key distribution $\mathcal{D}$ and let $\mathcal{A}$ be a PPT adversary interacting with* KE *via the queries defined in Section 2.1 within 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 $b_{\mathsf{test}} \overset{\$}{\leftarrow} \{0,1\}$ 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 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 1 and 2.*

3 Modeling Breakdown Resilience

In the following, we describe how to integrate *breakdown resilience* into the generic (Bellare–Rogaway-style) security model for authenticated key exchange.

3.1 Discussion

For breakdown resilience, 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. Figure 1 illustrates the kind of executions we cover in our model, as well as the cases that are excluded due to reasons we discuss in the following. Although one may still hope for some form of security for future sessions after a breakdown of a primitive, for general schemes one cannot expect any security guarantees anymore (since the primitive may be crucial). We are therefore interested in the question if the expected security level is still achieved in *past* sessions and thus exclude sessions that are still active at the time of breakdown (cf. Figure 1, Scenario 3).

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 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 4 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 4). We use the 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

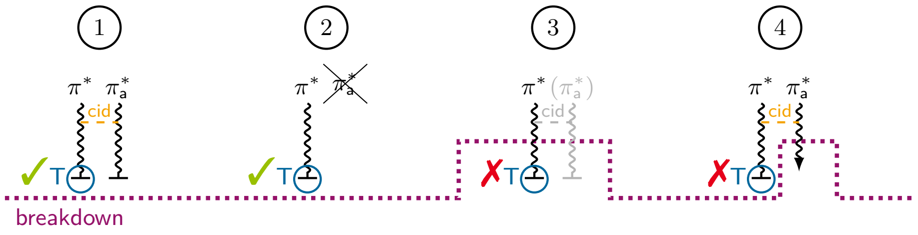


Fig. 1: Illustration of (non-)permissible Test queries wrt. a breakdown. The dotted line indicates the point in time of a breakdown with respect to the four scenarios of running or completed sessions; T denotes a test query on session π^*; π^*_{a} denotes an associated session (semi-)partnered with π^*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

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 a 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 $\mathcal{F}_{\mathsf{BDR}} = \{(f_1, \mathsf{sec\text{-}prop}_1), (f_2, \mathsf{sec\text{-}prop}_2), \dots\}$, 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.

Applicability to hybrid key exchange protocols. We note that the proposed model for breakdown resilience can be easily adjusted to cover the security analysis of so-called *hybrid* key exchange protocols. Hybrid key exchange protocols follow the idea of [37] and combine classically secure DH-based key exchange with post-quantum secure key exchange protocols. This approach allows the gradual introduction of post-quantum secure designs while still retaining the security of today's classical key exchanges. The security of hybrid key exchange protocols should thus be retained both for past and ongoing/future sessions, if only a single one of the two employed hardness assumption breaks. The breakdown resilience framework proposed in this paper can effortlessly handle the analysis of these protocols by not employing the modified version of the Send query and slightly adjusting the definition of BDR Key Secrecy.

3.2 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 $\mathsf{breakdown}$ (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 to the set associated with each session $\pi^k_{U,V}$:

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

[3] We here use the formalization by Dowling et al. [28] 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 [25] and the notion of (peer-)exchange variables used by Bhargavan et al. [10].

To avoid trivial choices and to relate the contributive identifiers to session identifiers we add two requirements for Match security: First, as in [28], 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 contributive identifier to prevent that Test queries are excluded by trivial choices of colliding contributive identifiers.

Break query. We add a Break query that complements the adversarial queries described in Section 2.1 and allows the adversary to schedule the timing of breakdowns. The query will set $\mathsf{breakdown}$ to true, record the current execution state of sessions, and provide 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 $\mathsf{sec\text{-}prop}$ 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 3.3 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.

$\mathsf{Break}()$: 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 $\underline{\text{or if } \mathsf{breakdown} = \mathsf{true}}$, then set $\mathsf{st}_{\mathsf{key}} \leftarrow \mathsf{revealed}$.

Primitive/Hardness Assumption (f)	**Algorithms**	**Security Assumption** (sec-prop)	Break **Response**
Encryption $\mathcal{E}$	$\mathcal{E} = (\mathsf{EKG}, \mathsf{Enc}, \mathsf{Dec})$	IND-CCA2 (indistinguishability under adaptive chosen ciphertext attack)	return all previous outputs (pk, sk) or sk for which $(pk, sk) \leftarrow \mathsf{EKG}$ or $sk \leftarrow \mathsf{EKG}$
Signatures $\mathcal{S}$	$\mathcal{S} = (\mathsf{SKG}, \mathsf{Sig}, \mathsf{SVf})$	EUF-CMA (existential unforgeability under chosen message attack)	return all previous pairs (pk, sk) for which $(pk, sk) \leftarrow \mathsf{SKG}$
MAC $\mathcal{M}$	$\mathcal{M} = (\mathsf{MKG}, \mathsf{MAC}, \mathsf{MVf})$	EUF-CMA (existential unforgeability under chosen message attack)	return all previous values sk for which $sk \leftarrow \mathsf{MKG}$
Hash Function $\mathcal{H}$	$\mathcal{H} = (\mathsf{HKG}, \mathsf{Hash})$	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
	$\mathcal{H} = (\mathsf{HKG}, \mathsf{RO})$	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 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 $k \leftarrow \mathsf{RO}(x)$
PRF $\mathcal{P}$	$\mathcal{P} = (\mathsf{PKG}, \mathsf{PRF})$	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	$\mathsf{GroupExp}(x) = g^{(x)}$ in multiplicative cyclic group $\mathbb{G} = \langle g \rangle$	Discrete Logarithm Problem	return all previous pairs (x, g^x) for which $g^x \leftarrow \mathsf{GroupExp}(x)$
Factoring	$\mathsf{GenModulus}(1^n) = (N, p, q)$ s.t. $N = p \cdot q$ where p, q are n-bit primes	Prime Factorization	return all previous tuples (N, p, q) for which $(N, p, q) \leftarrow \mathsf{GenModulus}(\cdot)$

Table 1: Break oracle specification.

3.3 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. In Table 1 we cover a wide range of standard primitives and assumptions underlying the security of most key exchange protocols (and in particular the NewHope [3] and TLS 1.3 [48] protocols we will analyze in Sections 4 and 5).

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 an encryption scheme $\mathcal{E} = (\mathsf{EKG}, \mathsf{Enc}, \mathsf{Dec})$ in honest sessions will be generated via the key generation algorithm EKG, 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 (identity function) 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 $\mathcal{A}$ must be able to generate collisions after the break. To this end, we allow $\mathcal{A}$ to program $\mathcal{H}$ globally on previously unseen input values after the breakdown occurred. In particular, 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 $\mathsf{GroupExp}$ 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 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 from the primitives in a specific key-exchange setting, one may wish to specify different responses for the Break query. Again, this is easily possible in our model as the Break query itself is generic.

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 $\mathsf{BDR\text{-}Match}$ security and BDR key secrecy. Both security notions differ from the original Bellare–Rogaway-like notions by the model 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 $\mathsf{BDR\text{-}Match}$ 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 4 ($\mathsf{BDR\text{-}Match}$ **Security).** *Let λ be the security parameter. Furthermore let KE be a key exchange protocol and $\mathcal{A}$ a PPT adversary interacting with KE via the queries $\mathsf{NewSession}$, $\mathsf{Send}_{\mathsf{BDR}}$, Reveal, $\mathsf{Corrupt}$, 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 $\mathsf{NewSession}$, $\mathsf{Send}_{\mathsf{BDR}}$, Reveal, $\mathsf{Corrupt}$, 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.)*
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* $\mathsf{BDR\text{-}Match}$-secure *for* $\mathcal{F}_{\mathsf{BDR}}$ *if for all PPT adversaries* $\mathcal{A}$ *the advantage function*

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

is negligible in the security parameter λ*.*

Definition 5 (BDR **Key Secrecy).** *Let* λ *be the security parameter. Furthermore let* KE *be a key exchange protocol with key distribution* $\mathcal{D}$ *and let* $\mathcal{A}$ *be a PPT adversary interacting with* KE *via the queries* $\mathsf{NewSession}$*,* $\mathsf{Send}_{\mathsf{BDR}}$*,* Reveal*,* $\mathsf{Corrupt}$*,* Break*, and* Test *within 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* $\mathsf{NewSession}$*,* $\mathsf{Send}_{\mathsf{BDR}}$*,* Reveal*,* $\mathsf{Corrupt}$*,* 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* $\pi'.\mathsf{st}_{\mathsf{exec}}^{\mathsf{bd}} = \mathsf{running}$*. (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* $G_{\mathsf{KE},\mathcal{A}}^{\mathsf{BDR}(\mathcal{F}_{\mathsf{BDR}}),\mathcal{D}}(\lambda) = 1$*, 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*

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

is negligible in the security parameter λ*.*

Definition 6 (Breakdown Resilience). *We say a key exchange protocol* KE *is* breakdown resilient for $\mathcal{F}_{\mathsf{BDR}}$ *(with/without forward secrecy) if* KE *provides* $\mathsf{BDR\text{-}Match}$ *security and* BDR *key secrecy for* $\mathcal{F}_{\mathsf{BDR}}$ *(with/without forward secrecy), according to Definitions 4 and 5.*

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 1. *If a key exchange protocol* KE *achieves breakdown resilience for any* $\mathcal{F}_{\mathsf{BDR}}$ *(incl.* $\mathcal{F}_{\mathsf{BDR}} = \emptyset$*) with/without forward secrecy according to Definition 6, then* KE *is also* BR*-secure with/without forward secrecy according to Definition 3.*

Proof. If the Break query is not asked by the adversary, the flag $\mathsf{breakdown}$ and the modification to the original Send query are essentially not touched and may thus be omitted. Likewise, the Finalize condition 2 in Definition 5 becomes void as $\mathsf{st}_{\mathsf{exec}}^{\mathsf{bd}} = \perp$ 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. $\square$

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 2. *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 straightforwardly 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 Break response for Y. $\square$

Remark 1. Perhaps surprisingly at first glance, breakdown resilience (for the according set of long-term secret primitives) does not imply forward secrecy. While both settings permit the exposure of long-term secrets after the tested session has accepted, breakdown resilience comes with slightly more restrictive (but necessary) conditions on when such exposure is permissible.

More precisely, Condition 2 in the BDR key secrecy definition (Definition 5) forbids a Test query to a session that had a contributive partner at the time of breakdown. In contrast, forward secrecy places no restrictions on when to corrupt a test session's owner beyond that the test session must have accepted before (a restriction that applies analogously to Break queries). This can result in a situation where a forward-secret $\mathsf{Corrupt}$ query enables an attacker to make two sessions derive the same key without being partnered, e.g., by re-randomizing a signature in a signed-Diffie–Hellman protocol where the signature enters the session identifier but not the key derivation. At the same time, the session deriving the same key as the test session may be contributively partnered to that session and hence, in the breakdown setting, the adversary's strategy—using a Break

query instead of the Corrupt to learn the long-term secrets—is not permissible. The example protocol hence would enjoy breakdown resilience for the employed signature scheme, but no forward secrecy.

4 NewHope

As a first application of our new security model, we analyze the breakdown resilience of an authenticated version of the (originally unauthenticated) NewHope protocol [3]. NewHope is a post-quantum secure key exchange protocol which was introduced in 2016 by Alkim et al. and has gained widespread attention, not least because of its experimental deployment in Google Chrome Canary [17]. The protocol's post-quantum security 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. NewHope improves the previous work by Bos et al. [15] with respect to efficiency and parameter size.

The efficiency improvement was to a great extent achieved by using errors e sampled from the centred binomial ψ_{16} instead of the usual discrete Gaussian χ. This results in a much more efficient sampling procedure with, as argued by the authors of [3], only a small loss in the security of the overall protocol. Furthermore, Alkim et al. made the scheme less susceptible to backdoors and all-for-the-price-of-one attacks by generating the usually fixed parameter a freshly in each protocol execution by applying SHAKE128 to a uniformly random seed seed at the outset of the protocol execution. SHAKE128 is a so-called *extendable-output function* (XOF) which allows for an input bit string to be extended to any desired output length. It was introduced alongside a higher-security variant SHAKE256 in the FIPS 202 SHA-3 standard [46].

4.1 The Auth-NewHope Protocol

The NewHope protocol has Alice and Bob exchange their RLWE public keys (b and b', respectively) and combines them in a Diffie–Hellman-like manner. However, due to the nature of the ring elements, Alice and Bob only agree on *approximately* the same value: Alice holds the value $v' = ass' + e's$, while Bob holds the value $v = ass' + es' + e''$. In order for them to agree on the common value w that will be the input to the final key derivation, they have to execute a so-called *error-reconciliation mechanism.* This is achieved by Bob providing additional *reconciliation information* from his value v via the function $\mathsf{HelpRec}(v)$ and sending the resulting value r to Alice. Both can then apply the reconciliation function $\mathsf{Rec}(\cdot,\cdot)$ to their noisy shared value with the reconciliation vector r to arrive at the exact shared value w with high probability. From this common shared value they then derive the session key via the function $\mathsf{KDF}(w)$; in NewHope the SHA3-256 function is used as KDF.

In its originally proposed form, NewHope provides unauthenticated key agreement. For our analysis, we consider an authenticated version of NewHope, in the following referred to as Auth-NewHope. The authenticated version of the protocol is depicted in Figure 2, where the original protocol (above the double line) is followed by a SIGMA-style authentication step as introduced by Krawczyk [39]. For details on the unauthenticated protocol, especially concerning the formal definition of the functions $\mathsf{HelpRec}(\cdot)$ and $\mathsf{Rec}(\cdot,\cdot)$, we refer the interested reader to the original protocol description in [3].

For illustrative purposes, the protocol description of Auth-NewHope here has been divided according to the two phases of the unauthenticated NewHope protocol 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.

4.2 Cryptographic Assumptions

Auth-NewHope relies on the following cryptographic primitives and hardness assumptions: random-oracle randomness of the extendable-output function XOF, pseudorandomness of the key derivation function KDF, hardness of the decisional Ring-LWE problem, 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.

With this, we can state the decisional version of the Ring-LWE problem:

Definition 7 **(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*

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

The key exchange part of Figure 2 is a modified version of the so-called *Decision Diffie–Hellman-like problem* ($\mathsf{DDH}\ell$):

Definition 8 **($\mathsf{DDH}\ell$ Problem).** *Let q, n, χ be Ring-LWE parameters. Given reconciliation information r, the* decision Diffie–Hellman-like *problem ($\mathsf{DDH}\ell$) for parameters q, n, χ asks to distinguish (a, b, b', r, w) from (a, b, b', r, w'), where $a \leftarrow \mathcal{U}(\mathcal{R}_q)$, $s, s', e, e'' \xleftarrow{\$} \chi$, $b \leftarrow as + e$, $b' \leftarrow as' + e'$, $v \leftarrow bs' + e''$, $r \xleftarrow{\$} \mathsf{HelpRec}(v)$, $w \leftarrow \mathsf{Rec}(v, r)$, and $w' \xleftarrow{\$} \mathcal{U}(\{0, 1\}^n)$. For an algorithm $\mathcal{A}$ define the distinguishing advantage to be*

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

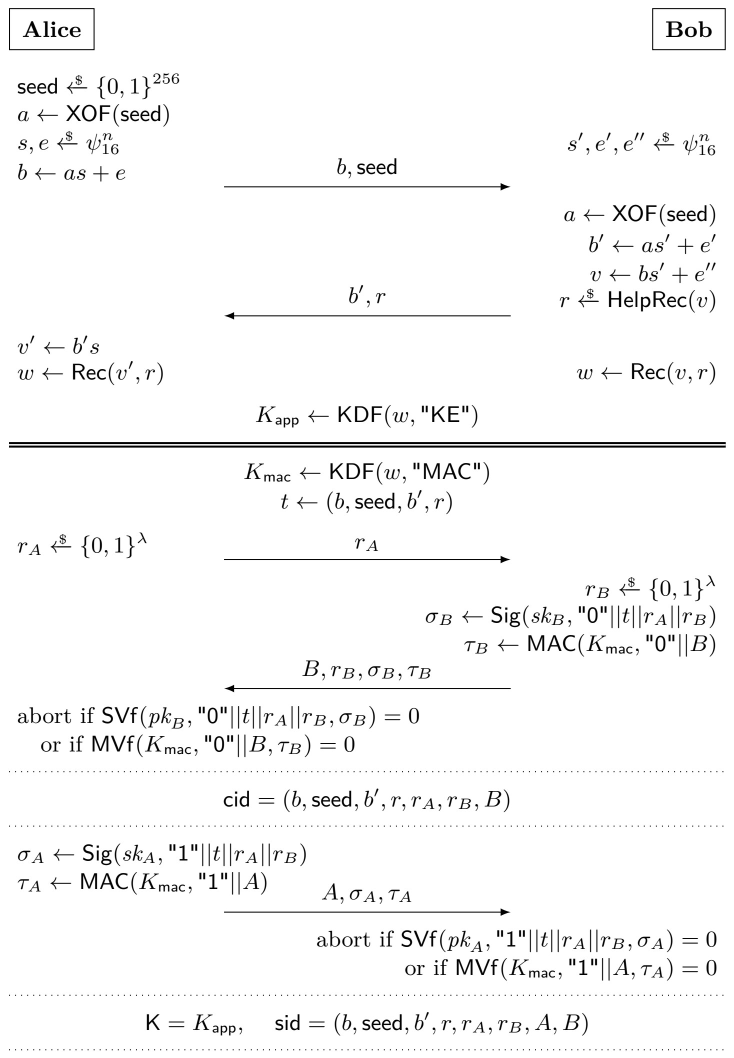

Alice
Bob
$\mathsf{seed} \xleftarrow{\$} \{0,1\}^{256}$
$a \leftarrow \mathsf{XOF}(\mathsf{seed})$
$s, e \xleftarrow{\$} \psi_{16}^{n}$
$s', e', e'' \xleftarrow{\$} \psi_{16}^{n}$
$b \leftarrow as + e$
b, seed
$a \leftarrow \mathsf{XOF}(\mathsf{seed})$
$b' \leftarrow as' + e'$
$v \leftarrow bs' + e''$
b', r
$r \xleftarrow{\$} \mathsf{HelpRec}(v)$
$v' \leftarrow b's$
$w \leftarrow \mathsf{Rec}(v', r)$
$w \leftarrow \mathsf{Rec}(v, r)$
$K_{\mathsf{app}} \leftarrow \mathsf{KDF}(w, \text{"KE"})$
$K_{\mathsf{mac}} \leftarrow \mathsf{KDF}(w, \text{"MAC"})$
$t \leftarrow (b, \mathsf{seed}, b', r)$
$r_A \xleftarrow{\$} \{0,1\}^{\lambda}$
r_A
$r_B \xleftarrow{\$} \{0,1\}^{\lambda}$
$\sigma_B \leftarrow \mathsf{Sig}(sk_B, \text{"0"}||t||r_A||r_B)$
$\tau_B \leftarrow \mathsf{MAC}(K_{\mathsf{mac}}, \text{"0"}||B)$
B, r_B, σ_B, τ_B
abort if $\mathsf{SVf}(pk_B, \text{"0"}||t||r_A||r_B, \sigma_B) = 0$
or if $\mathsf{MVf}(K_{\mathsf{mac}}, \text{"0"}||B, \tau_B) = 0$
$\mathsf{cid} = (b, \mathsf{seed}, b', r, r_A, r_B, B)$
$\sigma_A \leftarrow \mathsf{Sig}(sk_A, \text{"1"}||t||r_A||r_B)$
$\tau_A \leftarrow \mathsf{MAC}(K_{\mathsf{mac}}, \text{"1"}||A)$
A, σ_A, τ_A
abort if $\mathsf{SVf}(pk_A, \text{"1"}||t||r_A||r_B, \sigma_A) = 0$
or if $\mathsf{MVf}(K_{\mathsf{mac}}, \text{"1"}||A, \tau_A) = 0$
$\mathsf{K} = K_{\mathsf{app}}, \quad \mathsf{sid} = (b, \mathsf{seed}, b', r, r_A, r_B, A, B)$

Fig. 2: The AUTH-NEWHOPE protocol, with the original, unauthenticated NEWHOPE protocol [3] above the double line.

It turns out that if the decision Ring-LWE problem is hard, so is the decision Diffie–Hellman-like problem:

Theorem 1 (Hardness of $\mathsf{DDH}\ell$ Problem). *Let q be an odd integer, n some parameter and χ be a distribution on $\mathcal{R}_q$. It holds: if the decision Ring-LWE problem for q, n, χ is hard, then the DDH-like problem for q, n, χ is hard, i.e., there exist efficient adversaries $\mathcal{B}_1, \mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathsf{DDH}\ell}_{q,n,\chi,\mathcal{A}} \le \mathsf{Adv}^{\mathsf{DRLWE}}_{q,n,\chi,\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{DRLWE}}_{q,n,\chi,\mathcal{B}_2}.$$

The proof of the theorem can for example be found in [15].

4.3 Breakdown Resilience of Auth-NewHope

In the following, we show that Auth-NewHope is breakdown resilient for $\mathcal{F}_{\mathsf{BDR}} = \{(\mathsf{XOF}, \mathsf{RO\text{-}Rand}), (\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathcal{M}, \mathsf{EUF\text{-}CMA})\}$ with forward secrecy by establishing the corresponding $\mathsf{BDR\text{-}Match}$ security and BDR key secrecy. Note that $\mathcal{F}_{\mathsf{BDR}}$ neither contains the hardness assumptions DRLWE (and $\mathsf{DDH}\ell$) nor the key derivation function KDF. This is due to the fact that a break of any of these makes key secrecy impossible to achieve. Without the hardness of DRLWE (and hence $\mathsf{DDH}\ell$) we cannot replace the input to the key derivation function by a uniform random value in order to later argue indistinguishability. Furthermore, the break of KDF causes the adversary to see all previous outputs of the key derivation function, thus trivially enabling it to distinguish real from random keys.

Theorem 2 ($\mathsf{BDR\text{-}Match}$ security of Auth-NewHope).
Let $\mathcal{F}_{\mathsf{BDR}} = \{(\mathsf{XOF}, \mathsf{RO\text{-}Rand}), (\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathcal{M}, \mathsf{EUF\text{-}CMA})\}$. Then Auth-NewHope *is $\mathsf{BDR\text{-}Match}$-secure for $\mathcal{F}_{\mathsf{BDR}}$. For any efficient adversary $\mathcal{A}$ we have*

$$\mathsf{Adv}^{\mathsf{BDR\text{-}Match}(\mathcal{F}_{\mathsf{BDR}})}_{\text{A-NH},\mathcal{A}} \le n_s^2 \cdot \min\left\{c, 2^{-256}, 2^{|\mathsf{nonce}|}, 2^{-|r|}\right\},$$

where n_s is the maximum number of sessions, c is the negligible probability that the same small element is sampled from ψ_{16}^n twice, $|\mathsf{nonce}| = 256$ is the bit-length of the nonces r_A and r_B, and $|r|$ denotes the bit length of the reconciliation vector r.

Proof. In order to achieve $\mathsf{BDR\text{-}Match}$ Security, we need to show that the four conditions are satisfied (cf. Definition 4). Recall that the session identifiers are defined as $\mathsf{sid} = (b, \mathsf{seed}, b', r, r_A, r_B, A, B)$, containing public information only, and that the contributive identifiers are set as $\mathsf{cid} = (b, \mathsf{seed}, b', r, r_A, r_B, B)$.
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 pick, as responder, its random values s', e', e'', r, and r_B such that they collide or, as initiator, pick colliding random values seed, s, e, and r_A. This will only happen with probability at most $\min\{c, 2^{-256}, 2^{-|r|}, 2^{-|\mathsf{nonce}|}\}$. 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 3 (BDR **key secrecy of Auth-NewHope).**
Let $\mathcal{F}_{\mathsf{BDR}} = \{(\mathsf{XOF}, \mathsf{RO\text{-}Rand}), (\mathcal{S}, \mathsf{EUF\text{-}CMA}), (\mathcal{M}, \mathsf{EUF\text{-}CMA})\}$*. 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:*

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

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* r_A *and* r_B*.*

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 ($\text{"b"}||t||r_A||r_B$) 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 ($\text{"b"}||t||r_A||r_B$) earlier, only ($\text{"b'"}||t||r_A||r_B$) 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 w in the test session (and its associated session π^*_{a}) by a uniformly random value $\widetilde{w} \xleftarrow{\$} \{0,1\}^{256}$. If the adversary $\mathcal{A}$ can distinguish Game 5 from Game 4, then there exists an adversary $\mathcal{B}_2$ that can solve the $\mathsf{DDH}\ell$ problem as follows.
Algorithm $\mathcal{B}_2$ obtains a $\mathsf{DDH}\ell$ challenge $(\hat{a}, \hat{b}, \hat{b}', \hat{r}, \hat{w})$ and simulates the environment for $\mathcal{A}$ picking the according seed seed' upfront and programming the

random oracle modeling XOF such that $\mathsf{XOF}(\mathsf{seed}') = \hat{a}$ before any execution starts. In the predicted sessions π^* and π^*_{a} algorithm $\mathcal{B}_2$ then sets $a = \hat{a}$, $b = \hat{b}$, $b' = \hat{b}'$, and $r = \hat{r}$. Note that it is irrelevant for the argument if another honest session accidentally picks the same seed seed' and thus derives the same $\hat{a}$ since the non-uniqueness of the parameter does not affect the security of the protocol in terms of key secrecy. In fact, in many Ring-LWE-based KE schemes, this parameter is globally fixed upfront for all executions. Furthermore, a breakdown of XOF does not imply any advantage for the adversary in detecting the simulation (as the value $\hat{a}$ will appear to have been validly generated in an honest execution) or disturbing the programming of the random oracle (since the breakdown can only happen after the test session has been completed). Thus, when computing the keys K_{app} and K_{mac} in the two sessions, the given value $\hat{w}$ 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 $\hat{w}$ is genuine, then the simulation above is as in Game 4. If $\hat{w}$ 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 solve $\mathsf{DDH}\ell$ efficiently with non-negligible advantage. It follows that

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

Since $\mathsf{DDH}\ell$ 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 $\mathsf{running}$ 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 $\mathsf{Corrupt}$ 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 $\mathsf{KDF}(w, \text{"}\mathsf{KE}\text{"})$ ($b_{\mathsf{test}} = 1$). At this point, both cases $b_{\mathsf{test}} = 0$ and $b_{\mathsf{test}} = 1$ are indistinguishable for $\mathcal{A}$ since both keys are drawn independently and uniformly at random from $\{0,1\}^{256}$. 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:

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

$\square$

Remark 2. The security of the extendable-output function XOF (modeled as random oracle) does not enter into the security bound for key secrecy. We recall that in NewHope the function XOF is applied to a uniformly random chosen seed to generate the public parameter a freshly for each protocol execution.

This is done to avoid backdoors and all-for-the-price-of-one-attacks. However, the security of RLWE-based protocols does not rely on a being indistinguishable from random as it is in general public and fixed for all executions (cf. for example [15]). Furthermore, note that the KDF and XOF proposed in [3] both rely on the (pseudo-)randomness of $\mathsf{SHA3}$. In our analysis we treat these two primitives as independent and generic cryptographic building blocks such that a break of XOF does not imply a break in the key derivation function KDF (and vice versa). This result shows that the protocol can withstand breakdowns of XOF if KDF is based on a different primitive.

Remark 3. It may be surprising at first that the unforgeability of the signature and MAC scheme enter into the security bound of Theorem 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 version 1.3, which is currently being developed by the Internet Engineering Task Force (IETF) with the most recent draft version being draft-22 [49]. Although not specifically designed with breakdown resilience as a security goal in mind, the TLS 1.3 key exchange (the so-called handshake) can achieve resilience against the breakdown of some of its cryptographic components, as we will see.

TLS 1.3 (in draft-21) essentially specifies four 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 of earlier TLS 1.3 drafts [28,29,24,33,45,41,32,8,23]. 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. Maybe surprisingly at first glance, 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. While there are engineering reasons for this particular design, as we will discuss later, our analysis 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 [31] in order to capture breakdown resilience for the multiple keys derived in TLS 1.3 with varying authentication properties (see also [28,29]).

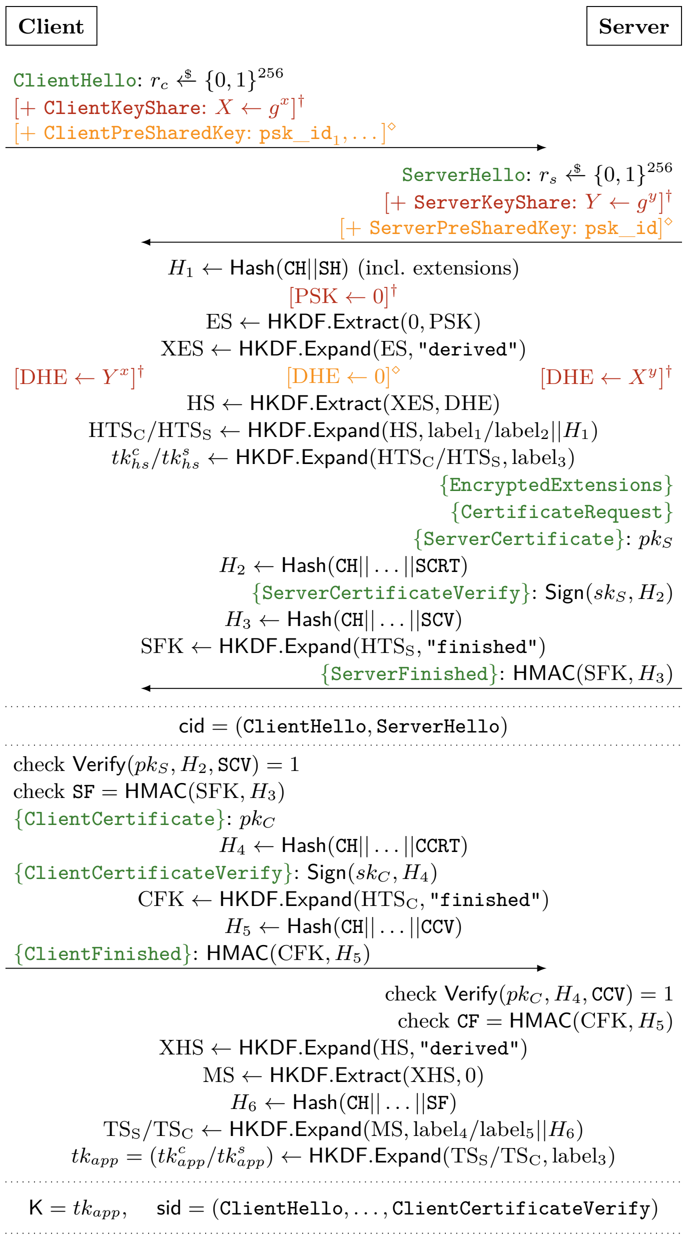
Full/(EC)DHE mode. We begin with explaining the full handshake mode based on (elliptic-curve) ephemeral Diffie–Hellman ((EC)DHE) key exchange. Figure 3 shows the TLS 1.3 handshake protocol flow; messages and computations marked with $[\dots]^\diamond$ are only included in the PSK-based handshake mode 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_{hs}^c and tk_{hs}^s. 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 [40] 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_{app}^c and tk_{app}^s) as the session key K.

[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 [5]: $\mathsf{HKDF.Extract}(XTS, SKM)$ 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).



Client
Server
ClientHello: $r_c \xleftarrow{\$} \{0,1\}^{256}$
[+ ClientKeyShare: $X \leftarrow g^x$]†
[+ ClientPreSharedKey: psk_id$_1, \ldots$]$^\diamond$
ServerHello: $r_s \xleftarrow{\$} \{0,1\}^{256}$
[+ ServerKeyShare: $Y \leftarrow g^y$]†
[+ ServerPreSharedKey: psk_id]$^\diamond$
$H_1 \leftarrow \mathsf{Hash}(\texttt{CH}||\texttt{SH})$ (incl. extensions)
$[\mathrm{PSK} \leftarrow 0]^\dagger$
$\mathrm{ES} \leftarrow \mathsf{HKDF.Extract}(0, \mathrm{PSK})$
$\mathrm{XES} \leftarrow \mathsf{HKDF.Expand}(\mathrm{ES}, \texttt{"derived"})$
$[\mathrm{DHE} \leftarrow Y^x]^\dagger$
$[\mathrm{DHE} \leftarrow 0]^\diamond$
$[\mathrm{DHE} \leftarrow X^y]^\dagger$
$\mathrm{HS} \leftarrow \mathsf{HKDF.Extract}(\mathrm{XES}, \mathrm{DHE})$
$\mathrm{HTS_C}/\mathrm{HTS_S} \leftarrow \mathsf{HKDF.Expand}(\mathrm{HS}, \mathrm{label}_1/\mathrm{label}_2||H_1)$
$tk^c_{hs}/tk^s_{hs} \leftarrow \mathsf{HKDF.Expand}(\mathrm{HTS_C}/\mathrm{HTS_S}, \mathrm{label}_3)$
{EncryptedExtensions}
{CertificateRequest}
{ServerCertificate}: pk_S
$H_2 \leftarrow \mathsf{Hash}(\texttt{CH}||\ldots||\texttt{SCRT})$
{ServerCertificateVerify}: $\mathsf{Sign}(sk_S, H_2)$
$H_3 \leftarrow \mathsf{Hash}(\texttt{CH}||\ldots||\texttt{SCV})$
$\mathrm{SFK} \leftarrow \mathsf{HKDF.Expand}(\mathrm{HTS_S}, \texttt{"finished"})$
{ServerFinished}: $\mathsf{HMAC}(\mathrm{SFK}, H_3)$
$\mathsf{cid} = (\texttt{ClientHello}, \texttt{ServerHello})$
check $\mathsf{Verify}(pk_S, H_2, \texttt{SCV}) = 1$
check $\texttt{SF} = \mathsf{HMAC}(\mathrm{SFK}, H_3)$
{ClientCertificate}: pk_C
$H_4 \leftarrow \mathsf{Hash}(\texttt{CH}||\ldots||\texttt{CCRT})$
{ClientCertificateVerify}: $\mathsf{Sign}(sk_C, H_4)$
$\mathrm{CFK} \leftarrow \mathsf{HKDF.Expand}(\mathrm{HTS_C}, \texttt{"finished"})$
$H_5 \leftarrow \mathsf{Hash}(\texttt{CH}||\ldots||\texttt{CCV})$
{ClientFinished}: $\mathsf{HMAC}(\mathrm{CFK}, H_5)$
check $\mathsf{Verify}(pk_C, H_4, \texttt{CCV}) = 1$
check $\texttt{CF} = \mathsf{HMAC}(\mathrm{CFK}, H_5)$
$\mathrm{XHS} \leftarrow \mathsf{HKDF.Expand}(\mathrm{HS}, \texttt{"derived"})$
$\mathrm{MS} \leftarrow \mathsf{HKDF.Extract}(\mathrm{XHS}, 0)$
$H_6 \leftarrow \mathsf{Hash}(\texttt{CH}||\ldots||\texttt{SF})$
$\mathrm{TS_S}/\mathrm{TS_C} \leftarrow \mathsf{HKDF.Expand}(\mathrm{MS}, \mathrm{label}_4/\mathrm{label}_5||H_6)$
$tk_{app} = (tk^c_{app}/tk^s_{app}) \leftarrow \mathsf{HKDF.Expand}(\mathrm{TS_S}/\mathrm{TS_C}, \mathrm{label}_3)$
$\mathsf{K} = tk_{app}$, $\mathsf{sid} = (\texttt{ClientHello}, \ldots, \texttt{ClientCertificateVerify})$

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}
$[\ldots]^\dagger$	message/computation only when including DHE
$[\ldots]^\diamond$	message/computation only when including PSK
a/b	alternative usage of a or b in analogous computation

Fig. 3: The TLS 1.3 draft-21 [48] handshake protocol (in full/(EC)DHE, PSK, and PSK-(EC)DHE mode).

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, 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 4 ($\mathsf{BDR\text{-}Match}$ 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*

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

[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.

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 BDR-Match security (cf. Definition 4) 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.

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 (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:*

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

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 $\mathsf{Corrupt}$ 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 [12].[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 [4,6] of the $\mathsf{HKDF.Extract}$ 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 HS $\leftarrow \mathsf{HKDF.Extract}(\ldots, \widetilde{\mathrm{DHE}})$ to its PRF oracle, hence simulating either Game 4 or Game 5. Thus,

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

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

[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., [42,29]) which employ the stronger pseudorandom-function oracle-Diffie–Hellman ($\mathsf{PRF\text{-}ODH}$) assumption [38,18] or the Gap-Diffie–Hellman assumption (in the random oracle model).

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

In the reduction, $\mathcal{B}_{11}$ uses its MAC oracle to compute the `ClientFinished` message computed with key $\widetilde{\mathrm{CFK}}$ 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 (draft-21) 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 breakdown resilience 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, and perhaps surprisingly at first glance, the PSK-only handshake mode in general 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 $\mathsf{HKDF.Expand}$, where—unlike in the (EC)DHE case with its per-session DH values—potentially the *same* pre-shared key PSK is used throughout multiple sessions. Consider an adversary $\mathcal{A}$ that, after running an honest protocol for the test session, breaks the collision resistance of Hash (via the Break query). It can then run another honest protocol execution using the same pre-shared key PSK (unknown to the adversary), programming Hash to yield hash values for the transcript of this session that collide with those computed in the test session.[8] The adversary may then reveal the later session which will derive 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 does not provide breakdown resilience against 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., [28,29,8]). We note that resilience against collision resistance breakdown of the hash function Hash could be achieved by using the *non-hashed* session transcript (or parts thereof, like the nonces) in the key derivation. However, for engineering reasons 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. [48]). Finally, when using pre-shared keys derived from the resumption master secret established in a prior full handshake, TLS 1.3 draft-21 suggests that such PSKs (issued via so-called tickets) should be used only once [48, Section 4.6.1], 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 DHE value added as key derivation input ensures that different sessions derive distinct session keys even under colliding hashed transcripts, following a similar argument to that employed in the analysis of the full (EC)DHE handshake (cf. Theorem 5). 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 PSK-(EC)DHE handshake of TLS 1.3 hence not only achieves

[8] To be precise, the adversary will target a collision in $H_6 \leftarrow \mathsf{Hash}(\mathtt{CH}\|\dots\|\mathtt{SF})$, included when deriving $\mathrm{TS}_\mathrm{S}/\mathrm{TS}_\mathrm{C}$. All other values on the way to tk_{app} are derived deterministically from PSK with fixed labels only.

[9] This aspect is reminiscent of the comment by Dowling et al. [28] on upstream hashing in the signatures sent in the TLS 1.3 (EC)DHE handshake.

forward secrecy (against PSK compromise) but also breakdown resilience (for the hash and MAC function employed).

6 Conclusion

We presented the first extension to a variant of the widely used Bellare–Rogaway model [7] 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 are confident that the presented ideas can also be integrated into other relevant models for authenticated key exchange, such as the CK model [21], its extension eCK [43], the ACCE model [38], as well as the multi-stage setting [31]. Moreover, the notion may even be transferred to different classes of cryptographic protocols. A particularly interesting direction is to apply the BDR model to the analysis of so-called *hybrid* key exchange protocols which combine two (or more) key exchange algorithms with differing hardness assumptions with the aim to achieve security even if one of them breaks.

References

1. D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Z. Béguelin, and P. Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *ACM CCS 15*, pages 5–17, 2015. (Cited on page 2.)
2. N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt. On the security of RC4 in TLS. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 305–320, 2013. (Cited on page 2.)
3. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum Key Exchange—A New Hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, 2016. (Cited on pages 2, 4, 6, 14, 19, 20, 21, and 27.)
4. M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *CRYPTO 2006*, pages 602–619, 2006. (Cited on page 33.)
5. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO'96*, pages 1–15, 1996. (Cited on page 28.)
6. M. Bellare and A. 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 33.)
7. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO'93*, pages 232–249, 1994. (Cited on pages 3, 6, and 37.)
8. K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *2017 IEEE Symposium on Security and Privacy*, pages 483–502, 2017. (Cited on pages 6, 27, and 36.)

9. K. Bhargavan, C. Brzuska, C. Fournet, M. Green, M. Kohlweiss, and S. Z. Béguelin. Downgrade resilience in key-exchange protocols. In *2016 IEEE Symposium on Security and Privacy*, pages 506–525, 2016. (Cited on page 6.)
10. K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and S. Zanella Béguelin. Proving the TLS handshake secure (as it is). In *CRYPTO 2014, Part II*, pages 235–255, 2014. (Cited on page 12.)
11. K. Bhargavan and G. Leurent. Transcript collision attacks: Breaking authentication in TLS, IKE and SSH. In *NDSS 2016*, 2016. (Cited on page 2.)
12. D. Boneh. The decision diffie-hellman problem. In *Algorithmic Number Theory: Third International Symposiun*, pages 48–63, 1998. (Cited on page 33.)
13. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé. CRYSTALS – kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634, 2017. `http://eprint.iacr.org/2017/634`. (Cited on page 5.)
14. J. W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In *ACM CCS 16*, pages 1006–1018, 2016. (Cited on page 2.)
15. J. W. Bos, C. Costello, M. Naehrig, and D. 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, 2015. (Cited on pages 2, 4, 19, 22, and 27.)
16. C. Boyd, Y. Cliff, J. G. Nieto, and K. G. Paterson. Efficient one-round key exchange in the standard model. In *ACISP 08*, pages 69–83, 2008. (Cited on page 5.)
17. M. Braithwaite. Google Security Blog: Experimenting with post-quantum cryptography. `https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html`, 2016. (Cited on pages 2, 4, and 19.)
18. J. Brendel, M. Fischlin, F. Günther, and C. Janson. PRF-ODH: Relations, instantiations, and impossibility results. In *CRYPTO 2017, Part III*, pages 651–681, 2017. (Cited on page 33.)
19. C. 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 8.)
20. C. Brzuska, M. Fischlin, B. Warinschi, and S. C. Williams. Composability of Bellare-Rogaway key exchange protocols. In *ACM CCS 11*, pages 51–62, 2011. (Cited on page 8.)
21. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT 2001*, pages 453–474, 2001. (Cited on pages 5 and 37.)
22. K. Cohn-Gordon, C. J. F. Cremers, and L. Garratt. On Post-compromise Security. In *IEEE 29th Computer Security Foundations Symposium (CSF 2016)*, pages 164–178, 2016. (Cited on page 5.)
23. C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *ACM CCS 17*, pages 1773–1788, 2017. (Cited on pages 6 and 27.)
24. C. Cremers, M. Horvat, S. Scott, and T. 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, 2016. (Cited on pages 6 and 27.)
25. C. J. F. Cremers and M. Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In *ESORICS 2012*, pages 734–751, 2012. (Cited on page 12.)

26. B. den Boer and A. Bosselaers. Collisions for the compressin function of MD5. In *EUROCRYPT'93*, pages 293–304, 1994. (Cited on page 2.)
27. W. Diffie, P. C. Van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992. (Cited on page 5.)
28. B. Dowling, M. Fischlin, F. Günther, and D. Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *ACM CCS 15*, pages 1197–1210, 2015. (Cited on pages 6, 12, 13, 27, 28, and 36.)
29. B. Dowling, M. Fischlin, F. Günther, and D. 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 6, 27, 28, 33, and 36.)
30. B. Dowling and D. Stebila. Modelling ciphersuite and version negotiation in the TLS protocol. In *ACISP 15*, pages 270–288, 2015. (Cited on page 6.)
31. M. Fischlin and F. Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In *ACM CCS 14*, pages 1193–1204, 2014. (Cited on pages 28 and 37.)
32. M. Fischlin and F. Günther. Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P 2017)*, pages 60–75, 2017. (Cited on pages 6 and 27.)
33. M. Fischlin, F. Günther, B. Schmidt, and B. 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, 2016. (Cited on pages 6 and 27.)
34. I. Giechaskiel, C. J. F. Cremers, and K. B. Rasmussen. On bitcoin security in the presence of broken cryptographic primitives. In *ESORICS 2016, Part II*, pages 201–222, 2016. (Cited on page 5.)
35. C. G. Günther. An identity-based key-exchange protocol. In *EUROCRYPT'89*, pages 29–37, 1990. (Cited on page 5.)
36. S. S. Gupta, S. Maitra, G. Paul, and S. Sarkar. (Non-)random sequences from (non-)random permutations - analysis of RC4 stream cipher. *Journal of Cryptology*, 27(1):67–108, 2014. (Cited on page 2.)
37. D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT 2005*, pages 96–113, 2005. (Cited on page 12.)
38. T. Jager, F. Kohlar, S. Schäge, and J. Schwenk. On the security of TLS-DHE in the standard model. In *CRYPTO 2012*, pages 273–293, 2012. (Cited on pages 33 and 37.)
39. H. Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *CRYPTO 2003*, pages 400–425, 2003. (Cited on pages 4 and 20.)
40. H. Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In *CRYPTO 2010*, pages 631–648, 2010. (Cited on page 28.)
41. H. Krawczyk. A unilateral-to-mutual authentication compiler for key exchange (with applications to client authentication in TLS 1.3). In *ACM CCS 16*, pages 1438–1450, 2016. (Cited on pages 6 and 27.)
42. H. Krawczyk and H. Wee. The OPTLS protocol and TLS 1.3. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P 2016)*, pages 81–96, 2016. (Cited on page 33.)

43. B. LaMacchia, K. Lauter, and A. Mityagin. *Stronger Security of Authenticated Key Exchange*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. (Cited on page 37.)
44. B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec 2007*, pages 1–16, 2007. (Cited on page 5.)
45. X. Li, J. Xu, Z. Zhang, D. Feng, and H. Hu. Multiple handshakes security of TLS 1.3 candidates. In *2016 IEEE Symposium on Security and Privacy*, pages 486–505, 2016. (Cited on pages 6 and 27.)
46. NIST. Federal Information Processing Standard 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, 2015. (Cited on page 19.)
47. K. G. Paterson and T. van der Merwe. Reactive and proactive standardisation of TLS. In *Security Standardisation Research (SSR 2016)*, pages 160–186, 2016. (Cited on page 6.)
48. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 – draft-ietf-tls-tls13-21. `https://tools.ietf.org/html/draft-ietf-tls-tls13-21`, 2017. (Cited on pages 3, 4, 14, 29, and 36.)
49. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 – draft-ietf-tls-tls13-22. `https://tools.ietf.org/html/draft-ietf-tls-tls13-22`, 2017. (Cited on page 27.)
50. M. Stevens. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In *EUROCRYPT 2013*, pages 245–261, 2013. (Cited on page 2.)
51. M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The first collision for full SHA-1. In *CRYPTO 2017, Part I*, pages 570–596, 2017. (Cited on page 2.)
52. M. Stevens, P. Karpman, and T. Peyrin. Freestart collision for full SHA-1. In *EUROCRYPT 2016, Part I*, pages 459–483, 2016. (Cited on page 2.)
53. M. Stevens, A. K. Lenstra, and B. de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In *EUROCRYPT 2007*, pages 1–22, 2007. (Cited on page 2.)
54. X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In *CRYPTO 2005*, pages 17–36, 2005. (Cited on page 2.)
55. X. Wang and H. Yu. How to break MD5 and other hash functions. In *EUROCRYPT 2005*, pages 19–35, 2005. (Cited on page 2.)

Supplementary Material

A Security Assumptions

Definition 9 ((Public Key) IND-CCA2 Security). *Let λ be the security parameter. Furthermore let $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme and let $\mathcal{A}$ be a PPT algorithm. We define the following* IND-CCA2 *security game $G^{\text{IND-CCA2}}_{\mathsf{Enc},\mathcal{A}}(\lambda)$:*

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 $c^* = \mathsf{Enc}(pk, m_b)$ 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

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

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 10 (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 11 (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 12 (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 13 (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).*