Flexible Authenticated and Confidential Channel Establishment (fACCE): Analyzing the Noise Protocol Framework

Benjamin Dowling[1], Paul Rösler[2], and Jörg Schwenk[2]

[1] Information Security Group, Royal Holloway, University of London
`benjamin.dowling@rhul.ac.uk`
[2] Horst-Görtz Institute for IT Security,
Chair for Network and Data Security, Ruhr University Bochum
`{paul.roesler,joerg.schwenk}@rub.de`

April 30, 2019

Abstract. The Noise protocol framework is a suite of channel establishment protocols, of which each individual protocol ensures various security properties of the transmitted messages, but keeps specification, implementation, and configuration relatively simple. Implementations of the Noise protocols are themselves, due to the employed primitives, very performant. Thus, despite its relative youth, Noise is already used by large-scale deployed applications such as WhatsApp and Slack. Though the specification describes and claims the security properties of the protocol patterns very precisely, there has been no computational proof yet. We close this gap.
Noise uses only a limited number of cryptographic primitives which makes it an ideal candidate for reduction-based security proofs. Due to its patterns' characteristics as channel establishment protocols, and the usage of established keys within the handshake, the authenticated and confidential channel establishment (ACCE) model (Jager et al. CRYPTO 2012) seems perfectly fit for an analysis of Noise. However, the ACCE model strictly divides protocols into two non-overlapping phases: the *pre-accept phase* (i.e., the channel establishment) and *post-accept phase* (i.e., the channel). Using the example of Noise, we show that this separation originates from the historic background of the TLS 1.2 proof, rather than it depicting the natural core of a channel establishment protocol. Similarly to TLS 1.3, Noise allows the transmission of encrypted messages as soon as *a* key is established (for instance, before any authentication between parties has taken place).
By proposing a generalization of the original ACCE model, we catch security properties of these *earlier* messages precisely. As our generalized model is aimed to capture security of multiple different channel establishment protocols, we then add flexibility to the security definition, comparable to the multi-stage key exchange model (Fischlin and Günther CCS 2014). We furthermore provide a broad discussion on the relations among and dimensions of the considered security properties as this plays a crucial role when defining security flexibly. Based on this, we observe that each message sent during the channel establishment can add new security properties, while inheriting those established in previous stages.
We give full security proofs for eight of the 15 basic Noise patterns to illustrate the flexibility and validity of this approach.

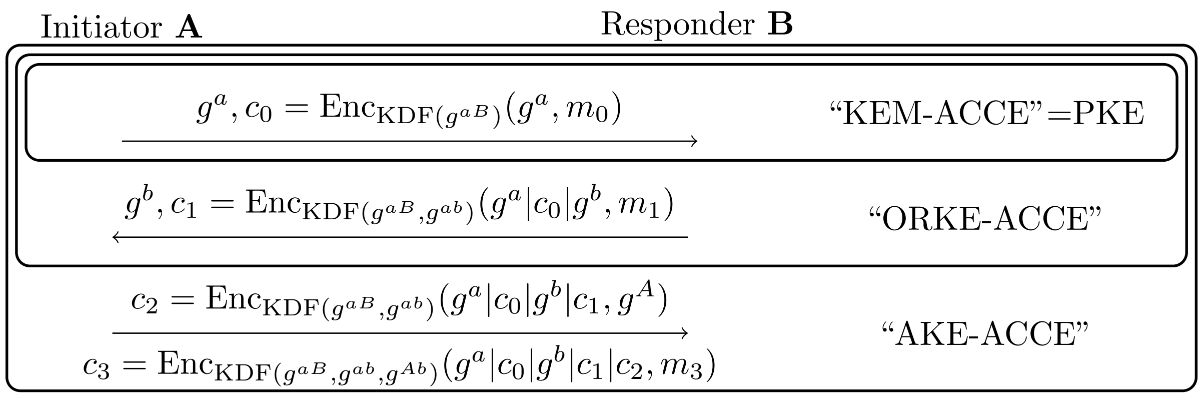
Table of Contents

1 Introduction

Noise is a protocol framework introduced by Trevor Perrin [38] for establishing confidential channels between two parties in various application scenarios, while using only a limited number of cryptographic primitives. Like TLS 1.2, Noise makes use of the derived keys during channel establishment (or handshake), which makes an analysis with respect to *key indistinguishability* as in traditional key exchange models infeasible. As with TLS 1.2, the Noise framework focuses on channel establishment and *not* on the establishment of a key, so the ACCE model [25] seems to capture the security of Noise perfectly. Many of the Noise protocols, however, do not separate between the handshake and the channel itself as strictly as modeled in ACCE. To allow the transmission of messages as early as possible (to avoid latency costs), protocols like TLS 1.3 and Noise amalgamate handshake and channel (at cost of security guarantees for these early messages). In this work, we show how to flexibilize the ACCE model, grasping its core idea and applying it to the Noise framework to prove fine-grained security guarantees.

The Noise Framework The Noise protocol framework is a tool box for defining simple and lightweight protocols for homogeneous environments. In this context, homogeneous means that all parties in the environment agree upon the protocol (including mechanisms for long-term key distribution, protocol version, employed cryptographic primitives, . . .). In contrast, TLS allows the establishment of a channel in highly federated environments, in which the information mentioned before is not previously known to protocol participants. This induces highly complex implementations that contain version and cipher suite negotiation as well as legacy code. Noise can disregard these issues (which in TLS regularly lead to security vulnerabilities, e.g., [1, 37]) but still offers multiple protocol patterns that allow a developer to choose a protocol fulfilling his or her application's security needs and considering the respective use case (long-term key distribution, latency, . . .).

The Noise specification defines 15 core protocol patterns for different use cases, which may consist of one, two, or three handshake messages (cf. Figure 1) – containing ephemeral and/or long-term Diffie-Hellman shares and (if a key is already established) an AEAD ciphertext – and a channel. Each party can have a long-term DH key pair, and potentially contributes one ephemeral DH key pair per protocol execution. The different patterns of Noise can hence be seen as different distributions of the corresponding two to four public DH shares to the handshake messages. The three-message patterns of Noise are novel in the sense that classical three/four-message patterns for AKE protocols typically use only one DH key exchange which is either static (TLS-DH) or ephemeral (signed DH, Station-to-Station protocol, TLS 1.3, TLS-DHE, IPsec IKE, SSH) combined with digital signatures (all of the above) or MACs (IPsec IKE Phase 2 with forward-secrecy). Noise avoids authentication with MACs or digital signatures, and provides entity authentication through long-term DH keys, key derivation, and AEAD.



Initiator **A**
Responder **B**
$g^a, c_0 = \mathrm{Enc}_{\mathrm{KDF}(g^{aB})}(g^a, m_0)$
"KEM-ACCE"=PKE
$g^b, c_1 = \mathrm{Enc}_{\mathrm{KDF}(g^{aB}, g^{ab})}(g^a|c_0|g^b, m_1)$
"ORKE-ACCE"
$c_2 = \mathrm{Enc}_{\mathrm{KDF}(g^{aB}, g^{ab})}(g^a|c_0|g^b|c_1, g^A)$
$c_3 = \mathrm{Enc}_{\mathrm{KDF}(g^{aB}, g^{ab}, g^{Ab})}(g^a|c_0|g^b|c_1|c_2, m_3)$
"AKE-ACCE"

Fig. 1: The flexible structure of the Noise protocol framework, described conceptually with the XK pattern (three passes) that bases on the NK pattern (two passes) that bases on the N pattern (one pass). g^A and g^B denote the long-term public DH shares of parties **A** and **B**, g^a and g^b denote their ephemeral shares, and $\mathrm{Enc}_k(ad, m)$ is an AEAD encryption.

As a result, Noise is for its scope even more agile than TLS, allowing tailored protocols for multiple use-cases with various security properties – making security analysis in a single static model difficult. Resulting from its efficiency and flexibility, Noise is used by largely deployed protocols such as WhatsApp [23, 42] (for client to server communication), Wiregurard [13, 15], Slack, Amazon AWS[1], and is potentially an ideal candidate for protecting the transport layer in IOT networks. Despite being distributed in applications used regularly by billions of users, there has not been a computational proof of Noise's security.[2]

Flexibility for ACCE Originally the ACCE model was developed with the strict separation between key establishment and communication channel in mind. The security of ACCE, however, does not require this separation, because it only targets on the confidentiality of transmitted messages and the authentication among communicating parties. Furthermore, the ACCE model as introduced requires mutual authentication for a channel. Krawczyk et al. [30] introduced a variant of the ACCE model, denoted SACCE, allowing them to analyze protocols with unilateral authentication. In addition, Lychev et al. [35] developed the two-stage ACCE model, denoted QACCE, to capture the security of QUIC. However, since QACCE is tailored specifically to the QUIC protocol, the model is barely applicable for the analysis of other protocols.

In order to generalize the idea of an ACCE model for the analysis of other channel establishment protocols then, it is important to overcome historic remnants of ACCE such as separation of channel establishment and channel, a dedicated session key[3], or strictly enforced authentication and security guarantees for *all* transmitted messages. In our model, we carefully focus on grasping the core idea of a channel establishment protocol in order to find a security notion that is not influenced by the specific protocol being analyzed.

Although the separation between handshake phase and channel phase is not a necessary property of channel establishment protocols, it is natural that a protocol that establishes a channel immediately (i.e., with the first protocol message) cannot fulfill the same security guarantees as protocols that take multiple round-trips before allowing the confidential transmission of payload. This intuition can be compared to different security levels that are achieved by key encapsulation mechanisms (KEM), one-round-key exchanges (ORKE), and authenticated key exchanges (AKE) as depicted in Figure 1. For example, one message patterns (i.e., KEM-DEM constructions) are, among other deficiencies, subject to replay attacks if not equipped with expensive key update mechanisms such as in [20]. As a result, such attacks must be considered when designing an appropriate security model. Our model takes these different stages of security goals into account by adding flexibility to the ACCE notion.

Since the Noise framework, on the one hand, directly complies with our syntax of a flexible ACCE and, on the other hand, defines multiple patterns with different, fine-grained security properties, we see it as an optimal candidate for verifying the agility of our generalization and flexibilization of the ACCE model.

Contributions Our contributions can be summarized as follows:

– We generalize and flexibilize ACCE by finding its core idea and removing remnants of historic constructions (such as a separation between handshake and channel, a single dedicated session key, or required authentication goals).
– We prove flexible ACCE security for the majority of Noise framework's standard protocol patterns, considering multiple fine-grained security properties of patterns. By focusing on the security of the established channels instead of the established session keys, this allows us to comprehend security claims of the Noise specification. Due to limited cryptographic primitives and similar patterns in Noise, this results in clear proofs.
– We thereby propose a comprehensible methodology to analyze channel establishment protocols with multiple stages, fulfilling different security properties.
– As modeling and defining security generically and flexibly requires the considered security properties to be well-understood, we provide a broad discussion on the interplay and dimensions of confidentiality and authenticity. In Appendices B and C we shed a light on the relations among authentication

[1] Both Slack and AWS use it in internal server-to-server communication.
[2] Except for the single pattern that is employed in the Wireguard protocol [15, 34].
[3] Think of a protocol that enforces the separation of the direction of the channel by using public key encryption instead of a symmetric key; hence a symmetric key is not necessarily part of the channel. We generalize it to a session state of each session participant.

and KCI resistance as well as attacks against long-term secrets and the ephemerally utilized randomness, which is also valuable for related primitives such as key exchange.

1.1 Related Work

Computational security proofs for real world protocols have a long history (e.g., [12, 14, 15, 16, 30, 36]). As described earlier, due to the usage of the channel key in the handshake of TLS 1.2, the ACCE model was introduced by Jager et al. [26] (which was later also used in [3, 6, 7, 8, 33]) as a proof of key indistinguishability was impossible without considering a modified protocol variant. To further analyze the security of TLS 1.2 without client authentication, Krawczyk et al. [30] and Kohlar et al. [29] independently proposed a variant of the ACCE model.

The issue of analyzing real world protocols that use the established key within the handshake has been bypassed in the past by modifying the protocol for the analysis [12, 15, 36]. While a certain fraction in the key exchange community is of the opinion that ACCE is the lesser alternative to modeling and analyzing protocols that do not carefully divide handshake and channel, we see it as an opportunity to elaborate accurate and precise statements on the security of communicated payload within a channel.

The multi-stage key exchange model by Fischlin and Günther [16] extends the Bellare-Rogaway model [2] (further extended by [14, 17]) similarly as we extend the original ACCE model. In addition to the issue of key-usage during the handshake in Noise (as in TLS 1.2 or Signal), the multi-stage key exchange model is too restrictive to be applied here: Authentication is only modeled flexibly as far as necessary for the respective analyzed protocols (QUIC in [16], TLS 1.3 in [14, 17]. Furthermore, extended security properties such as Key Compromise Impersonation (KCI) resistance are not captured.

Giesen et al. [18] extended the ACCE model to consider multiple stages during a protocol execution to analyze TLS renegotiation. Besides its static security definition(s) and in addition to inheriting other unnecessary remnants of the ACCE model, all stages necessarily consist of separate handshake and channel phases (making it unapplicable for generic multi-stage protocols). Another step towards considering stages in ACCE was taken by Lychev et al. [35] and more recently by Chen et al. [11]. Their QACCE and msACCE models are, however, strongly tailored to the respectively analyzed protocols (QUIC and TLS 1.3). Blazy et al. [4] also proposed very recently a multistage ACCE model to analyze a ratcheting protocol (related to Signal). Similarly, their model strongly depends on the analyzed protocol, pursuing a contrary strategy to ours (i.e., a specialized instead of a generic model).

Previous to our work, Dowling and Paterson [15] examined the WireGuard key exchange protocol [13], itself based upon a single variant of Noise called pattern `IKpsk2`. They show that analyzing WireGuard in a key-indistinguishability-based security framework is impossible, as the protocol relies on an encrypted message using the established session keys to act as a key-confirmation message. They instead modify the WireGuard key exchange protocol to morally capture the key confirmation message, and prove the modified construction secure. Recently Lipp et al. [34] confirmed the security of the WireGuard protocol by an automated analysis with CryptoVerif. Using this tool, they were able to produce a computational proof of security. Independently and concurrent to our work, Kobeissi and Bhargavan [27, 28] published a framework for the formal verification (and automatic code generation) of Noise patterns. In particular, they formalize Noise patterns and use transition logic to create symbolic models of dynamically chosen Noise patterns to allow automatic verification using ProVerif. This is a strong indication for Noise's security but the approach and the results can barely be compared with computational, reduction-based proofs with respect to generic security models. As their verification of all base Noise patterns is conducted automatically with respect to the security statements from the Noise specification and we provide a reduction-based proof of security in a generalized, flexible computational model manually, we see these two approaches to be complementary.[4]

[4] Please note that symbolic analyses disregard the actual representation of the cryptographic building blocks' input and output values. Thus, in symbolic analyses, cryptographic primitives are highly idealized. Consequently, while reduction-based proofs provide relations to well studied hardness assumptions, symbolic analyses assume "unconditional" security of a protocol's building blocks. Nevertheless, automatic proofs are less error-prone and better scalable.

2 Preliminaries

Here we formalize the notation and provide intuitions for security assumptions that we will utilize in our analysis of the Noise Protocol Framework. Standard assumptions and security notions such as *collision resistance* for hash functions, security of *pseudo-random functions*, and the full definition of the *PRF-Oracle-Diffie-Hellman* assumption can be found in Appendix A.

2.1 Notation

The following notation will be used throughout the paper. For $q \in \mathbb{N}$ by $[q]$ we denote the set $\{1, \cdots, q\}$. For a function $F : \{0,1\}^a \rightarrow \{0,1\}^b$, a describes the input length and b describes the output length of the function. If a or b take the value $*$ we say that the function is defined for inputs or outputs of arbitrary length. Let S be a finite set and let $|S|$ be its size. We say a value x is chosen uniformly at random by $x \leftarrow_{\$} S$. Let $\mathcal{A}$ be a probabilistic algorithm, we let $y \leftarrow_{\$} \mathcal{A}(x_1, ...)$ denote running $\mathcal{A}$ on input $(x_1, ...)$ with uniformly chosen random coins, and assigning the output to y. If $\mathcal{A}$ is a deterministic algorithm, then $y \leftarrow \mathcal{A}(x_1, ...)$ denotes that y is computed by $\mathcal{A}$ using $(x_1, ...)$ as input. By $y \leftarrow_{[r]} \mathcal{A}(x_1, ...)$ we denote that a probabilistic algorithm $\mathcal{A}$ is invoked deterministically by *consuming* its random coins from r. *Consumed* random coins are afterwards removed from r.[5] ϵ is the empty string and $\perp$ is a special element indicating no input or no output.

2.2 The PRF-Oracle-Diffie-Hellman Assumption

The $\mathsf{PRF\text{-}ODH}$ assumption was introduced by Jager et al. [25]. Brendel et al. [5] published a generalization of the $\mathsf{PRF\text{-}ODH}$ assumption and its variants. In our proof we will use the generic $\mathsf{PRF\text{-}ODH}$ assumption as defined in Appendix A. Intuitively, the $\mathsf{PRF\text{-}ODH}$ assumption says that a PRF, which is keyed by a value derived from a DH key exchange of two DH shares, outputs a pseudo random value even though an adversary has oracle access to the PRF under different combinations of these DH shares.

3 The Noise Protocol Framework

The Noise Protocol Framework (hereafter referred to as "Noise") is a specification that describes a framework with which two party channel establishment protocols can easily be instantiated for multiple purposes. The core of the framework is represented by the definition of 15 base protocol patterns, the exact descriptions of which can be found in Appendix F. Each of these patterns employs only four underlying cryptographic primitives: a Diffie-Hellman group, a hash function, a key derivation function, and an AEAD cipher. Depending on how these cryptographic primitives are combined, the channel establishment protocols achieve different cryptographic properties. The main properties are:

1. Authentication and integrity,
2. Key compromise impersonation (KCI) resistance,
3. Forward-secrecy, and
4. Resistance against replay attacks.

Another interesting security property that is achieved by the protocols, but not explicitly claimed, is:

5. Resistance against reveals of random coins of the session participants.

The 15 patterns mainly differ in the setup in which they can be deployed. There are patterns that do not require the initial distribution of users' long-term public keys (and either insist on the authentication of users by transmitting these keys either in plaintext or encrypted, or alternatively to disregard authentication altogether), and patterns that are based on the previous distribution of users' public keys. The out-of-band mechanism for public-key distribution is outside the scope of the specification, but one can imagine scenarios in which these keys are manually configured, can be acquired from a trusted third party, or are shipped with the respective application that uses Noise.

[5] In common notation this short cut would be denoted by: $r^* \| r \leftarrow r$; $y \leftarrow \mathcal{A}(x_1, ...; r^*)$.

While historic protocols strictly separated key establishment and channel, recent specifications (such as TLS 1.3) also allow these phases to be interleaved. This allows the early transmission of payload data but results in reduced – and perhaps staged – levels of security for this data. The Noise specification provides a detailed description of security properties for the data transmission in each round-trip of the handshake and for the channel of each pattern.

While a key feature of Noise is the omission of a negotiation of a pattern or the negotiation of the exact employed cryptographic algorithms (in contrast to TLS, Noise is intended to be used in settings in which all participants are configured equally), recent discussions on the mailing list consider negotiation as a feature in the future[6] – which we will not regard in our analysis. Also outside the scope of our analysis, Noise allows further features such as symmetric pre-shared keys.

Implementation Assumptions The Noise specification provides suggestions for some implementation details (but does not mandatorily require them). For our analysis, we assume that the protocol implementation follows these suggestions:

– No padding is employed (i.e., the length of the plaintext message is the same as the length of the encrypted message), and
– If an algorithm is called irregularly (an initiator receives before sending once, a party waits for ciphertext but encryption is invoked, decryption fails, ...), then the respective algorithm outputs an empty state and aborts.

3.1 Noise Protocol Patterns

Here we explain the details of Noise, necessary to understand the core protocols and their properties.

A pattern is defined by the knowledge of each participant regarding the long-term public key (or static public key) of the respective partner (before the handshake and during the handshake). For unidirectional (i.e., one-message) patterns, the single letter of the pattern name indicates whether the initiator's long-term public key is <u>n</u>ot defined (`N`), trans(<u>X</u>)mitted during the handshake (`X`), or <u>k</u>nown by the receiver in advance (`K`). It is clear that, for unidirectional patterns, the receiver's long-term public key needs to be known by the initiator in advance since otherwise no payload can be encrypted to the receiver. In the two-letter names of interactive patterns, the first letter indicates whether the initiator's long-term public key is not defined, X-mitted, or known by the responder, and the second letter indicates the same for the responder towards the initiator. So in the `XK` pattern, the initiator knows the responder's long-term public key in advance and the responder obtains the initiator's long-term public key during the handshake. At the top of Figure 2 (in which we depict three example Noise patterns) it is denoted that the initiator knows the responder's long-term public key and the responder knows its long-term secret key for patterns `N` and `NK` a priori. For pattern `XK`, the initiator additionally knows its own secret key (for which the public key is sent to the responder during the protocol execution).

Finally, the Noise specification distinguishes whether the long-term public key is sent in plaintext or encrypted (for the former, the letter would be `I` instead of `X`). The specification defines all pairwise letter-combinations among the three variants `N`, `X`, `K`, the unidirectional patterns `N`, `X`, `K`, and the three variants in which the initiator sends its long-term DH share in plaintext (i.e., `I_`).

The handshake of a Noise pattern always starts with the initialization of the local state. This local state contains:

1. a boolean that indicates the session's role (initiator/responder),
2. the pattern name,
3. potentially the session owner's long-term DH exponent and DH share (X, g^X),
4. potentially the intended partner's long-term public DH share g^Y,
5. potentially the session's ephemeral DH exponent and DH share (x, g^x),
6. potentially the peer's ephemeral public DH share g^y,
7. the chaining key ck,
8. the hash variable h,
9. the key(s) k (or $k_{\mathtt{i}}, k_{\mathtt{r}}$ for the channel), and
10. the nonce(s) n (or $n_{\mathtt{i}}, n_{\mathtt{r}}$).

[6] `https://moderncrypto.org/mail-archive/noise/2018/001495.html`

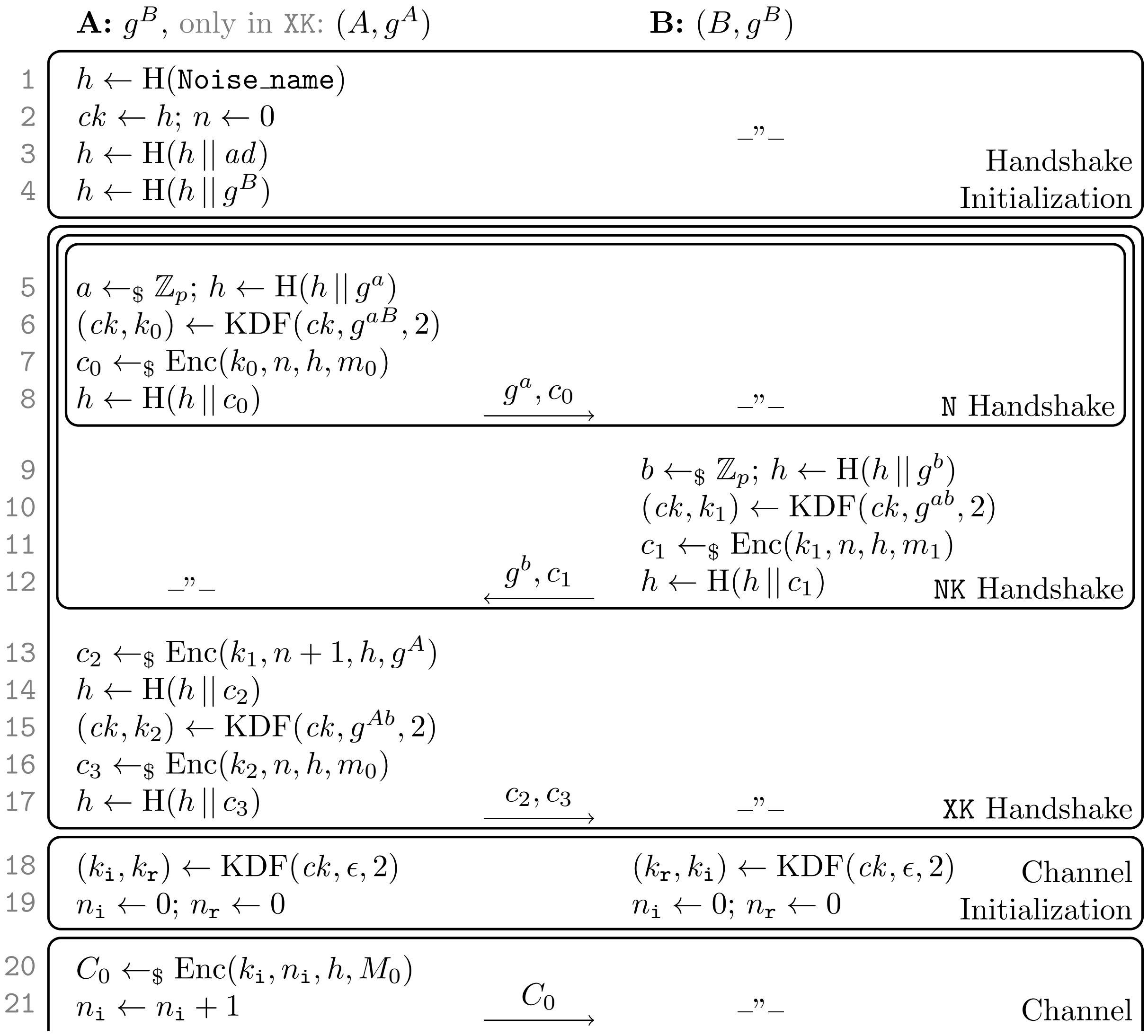

A: g^B, only in XK: (A, g^A)
B: (B, g^B)
1 $h \leftarrow \mathrm{H}(\texttt{Noise\_name})$
2 $ck \leftarrow h$; $n \leftarrow 0$
3 $h \leftarrow \mathrm{H}(h \,||\, ad)$
4 $h \leftarrow \mathrm{H}(h \,||\, g^B)$
–”–
Handshake Initialization
5 $a \leftarrow_\$ \mathbb{Z}_p$; $h \leftarrow \mathrm{H}(h \,||\, g^a)$
6 $(ck, k_0) \leftarrow \mathrm{KDF}(ck, g^{aB}, 2)$
7 $c_0 \leftarrow_\$ \mathrm{Enc}(k_0, n, h, m_0)$
8 $h \leftarrow \mathrm{H}(h \,||\, c_0)$
$\xrightarrow{g^a, c_0}$
–”–
N Handshake
9 $b \leftarrow_\$ \mathbb{Z}_p$; $h \leftarrow \mathrm{H}(h \,||\, g^b)$
10 $(ck, k_1) \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
11 $c_1 \leftarrow_\$ \mathrm{Enc}(k_1, n, h, m_1)$
12 –”– $\xleftarrow{g^b, c_1}$ $h \leftarrow \mathrm{H}(h \,||\, c_1)$
NK Handshake
13 $c_2 \leftarrow_\$ \mathrm{Enc}(k_1, n+1, h, g^A)$
14 $h \leftarrow \mathrm{H}(h \,||\, c_2)$
15 $(ck, k_2) \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2)$
16 $c_3 \leftarrow_\$ \mathrm{Enc}(k_2, n, h, m_0)$
17 $h \leftarrow \mathrm{H}(h \,||\, c_3)$
$\xrightarrow{c_2, c_3}$
–”–
XK Handshake
18 $(k_\mathtt{i}, k_\mathtt{r}) \leftarrow \mathrm{KDF}(ck, \epsilon, 2)$ $(k_\mathtt{r}, k_\mathtt{i}) \leftarrow \mathrm{KDF}(ck, \epsilon, 2)$
19 $n_\mathtt{i} \leftarrow 0$; $n_\mathtt{r} \leftarrow 0$ $n_\mathtt{i} \leftarrow 0$; $n_\mathtt{r} \leftarrow 0$
Channel Initialization
20 $C_0 \leftarrow_\$ \mathrm{Enc}(k_\mathtt{i}, n_\mathtt{i}, h, M_0)$
21 $n_\mathtt{i} \leftarrow n_\mathtt{i} + 1$
$\xrightarrow{C_0}$
–”–
Channel

Fig. 2: Fully specified N,NK and XK patterns. m_i are payload messages sent *during* the handshake; M_i are payload messages sent *after* the handshake; ad is associated data with which the handshake is initiated; –”– denotes that the respective operations for receipt are processed. Handshake initialization, channel initialization, and channel are part of all patterns.

These values are set, considering the pattern name, associated data ad, and a priori known long-term public DH shares of the partners (see Figure 2 lines 1-4).

For each handshake message (note that we refer to these messages as ciphertexts hereafter) the following operations can be conducted:

(a) the generation of an ephemeral DH exponent and the transmission of the respective DH public share,
(b) the plain or encrypted transmission of a long-term DH share,
(c) the computation of a DH secret from a public DH share of the partner and their own DH exponent.

The actual operations in the protocol for operation (a) are 1. the sampling of a DH exponent, 2. the hashing of its public share into h, and 3. the transmission of this public share to the partner (lines 5,9). In case (b), the sender's long-term DH share is encrypted under the current key k and the ciphertext is hashed into h and sent to the partner (lines 13-14). For patterns in which the long-term DH share is sent in plaintext, this DH share is hashed into h instead. If (c) a DH secret value was computed, the current ck together with the DH secret value, computed using the initiator's $\kappa_\mathtt{i}$ key and the responder's

$\kappa_{\mathtt{r}}$ key (where $\kappa_{\mathtt{i}}$ and $\kappa_{\mathtt{r}}$ are one of the ephemeral and long-term DH share each), is given as input to an invocation of KDF (lines 6,10,15).

For each handshake ciphertext in which a key k was already computed, a ciphertext under this current key k is derived by encrypting a payload m or (if no payload exists yet) an empty string ϵ. This ciphertext is sent to the partner and also hashed into h. The current value of h is associated data for every encryption (lines 7-8,11-12,16-17).

After all handshake ciphertexts are processed, the channel is initialized with a symmetric key for each direction, derived by invoking the KDF on the current chaining key ck (lines 18-19). In one-message patterns such as `N`, payload can however only be sent in one direction.

Flexibility in `N`, `NK`, `XK` Figure 2 depicts the three Noise patterns `N`, `NK`, and `XK`. As it can be seen, the `XK` pattern adds one further handshake ciphertext to the `NK` pattern such that the initiator is authenticated, and the `NK` pattern adds one handshake ciphertext to the `N` pattern, such that the responder is authenticated and a bidirectional forward secure channel is established.

4 Flexible ACCE Framework

The original ACCE model [25] and our generalization focus on the definition of *authentication* and *confidentiality* of messages, transmitted via a bidirectional communication protocol (*channel establishment*). However in [25], traditional security goals like authentication and forward-secrecy are required to be reached before the actual channel is established. This requirement seems to originate from the specific protocol for which ACCE was designed, Transport Layer Security (TLS 1.2).

Here we first provide a generic definition of the cryptographic primitive fACCE, then describe the standard execution environment in which its security is analyzed, further explain how we add flexibility to the adversary model with respect to the considered security properties, and finally define security of fACCE.

4.1 ACCE Primitive Description

In the following we provide a definition of the primitive *flexible ACCE*. Intuitively, it is a protocol that establishes a secure channel. Both, the establishment of the channel and the transmission of payload through the channel, are handled by the same algorithms[7]. The special 'security level'-output of encryption and decryption signals which security properties are reached by the current algorithm invocation (e.g., to a higher level application). We further explain this below.

Definition 1 (Flexible ACCE). *A flexible ACCE protocol* fACCE *is a tuple of algorithms* $\mathrm{fACCE} = (\mathrm{KGen}, \mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$ *defined over a secret key space* $\mathcal{SK}$*, a public key space* $\mathcal{PK}$*, and a state space* $\mathcal{ST}$*. The syntax of an* fACCE *protocol is as follows:*

- $\mathrm{KGen} \rightarrow_{\$} (sk, pk)$ *generates a long-term key pair where* $sk \in \mathcal{SK}, pk \in \mathcal{PK}$.
- $\mathrm{Init}(sk, ppk, \rho, ad) \rightarrow_{\$} st$ *initializes a session to begin communication, where sk (optionally) is the caller's long-term secret key, ppk (optionally) is the long-term public key of the session partner,* $\rho \in \{\mathtt{i}, \mathtt{r}\}$ *is this session's role (i.e., initiator or responder), ad is data associated with this session, and* $sk \in \mathcal{SK} \cup \{\bot\}, ppk \in \mathcal{PK} \cup \{\bot\}, ad \in \{0,1\}^*, st \in \mathcal{ST}$.
- $\mathrm{Enc}(sk, st, m) \rightarrow_{\$} (st', c, \varsigma)$ *continues the protocol execution in a session and takes message m to output new state* st'*, ciphertext c, and stage* ς *that indicates the security for the transmission via c of the input message, where* $sk \in \mathcal{SK} \cup \{\bot\}, st, st' \in \mathcal{ST}, m, c \in \{0,1\}^*, \varsigma \in \mathbb{N}$.
- $\mathrm{Dec}(sk, st, c) \rightarrow_{\$} (st', m, \varsigma)$ *processes the protocol execution in a session triggered by c and outputs new state* st'*, message m, and stage* ς *that indicates the security for the output message during transmission via c, where* $sk \in \mathcal{SK} \cup \{\bot\}, st \in \mathcal{ST}, st' \in \mathcal{ST} \cup \{\bot\}, m, c \in \{0,1\}^*, \varsigma \in \mathbb{N}$. *If* $st' = \bot$ *is output, then this denotes a rejection of this ciphertext.*

[7] One could further imagine that updates of the channel state, for reaching security properties such as forward-secrecy or post-compromise security, are processed by these algorithms.

Notes on the Syntax Our syntax differs from previous approaches to ACCE protocols that originated from notions of composition. We instead see fACCE as a primitive that is potentially built from authenticated key exchange (AKE) and secure channel protocols, but not necessarily *cleanly separated* into the "pre-accept" phase that establishes secrets and a "post-accept" phase that securely transmits payloads. We directly model all communication (handshake and payload transmission) via algorithms Enc and Dec which not only capture the secure channel but also *handshake operations* for the channel establishment. As the bytes sent over the network do not need to be further specified, we simply call them *ciphertexts* even though payload is not necessarily encrypted. We similarly view a *single* dedicated session key as a legacy of instantiating ACCE protocols via the composition of AKE and channel protocols. Since there are ways to secure the transmission of payload data other than simply using a symmetric key – consider asymmetric channels that use public key cryptography – we entirely subsume session-specific information in the session state. To capture unilateral-authenticated or unauthenticated establishment of a secure channel, not every participant of a session needs to use a long-term key pair. Furthermore, as ACCE protocols are not specifically aimed to hide the length of payload, we drop length-hiding properties from our syntax (see [25]).

Finally, we assume that the protocol outputs a *stage number* ς with every encryption and decryption. This ς indicates the 'security level' of the transmitted message (e.g., towards an upper layer application). In the case of an ACCE protocol in which all security properties are reached at once, this stage number is equivalent to distinguishing between the pre- and post-accept phase. In case of multi-stage protocols, a security classification can be useful information for an upper layer application that can then decide when to transmit confidential content. Since there exists no other indication to differentiate multiple stages based on our syntax[8], it is essentially necessary for defining security that the protocol itself outputs the stage numbers. We define as a convention that for output stage numbers $\varsigma = 0$ no security properties (especially no confidentiality) for the respectively transmitted payload is reached.

Please note that the syntax (and our security definition) leaves it to the specific protocol how far it enforces a ping-pong communication within a session. If the protocol only allows encryptions after decryptions, then we assume that the protocol enforces this by aborting on invalid algorithm invocations. If the protocol automatically responds on received ciphertexts, we assume that the environment (in our security experiment this is depicted by the adversary) handles this. Furthermore, we only consider protocols with FIFO channels (i.e., enforcing correct message order and prohibiting message omissions). In Appendix C.1 we describe how to extend our model to capture protocols that tolerate unreliable underlying networks.

We define the correctness of an fACCE protocol below. Intuitively an fACCE protocol is correct if messages, decrypted and received through the established channel, were equally sent to this channel by the partner. We assume without loss of generality that the output stage numbers monotonically increase during a session. In Appendix D Figure 7 we define correctness in a pseudocode notation to provide more clarity.

Definition 2 (Correctness of fACCE**).** *An* fACCE *protocol is correct if, for some* $(sk_{\mathtt{i}}, pk_{\mathtt{i}}), (sk_{\mathtt{r}}, pk_{\mathtt{r}}) \in ((\mathcal{SK} \times \mathcal{PK}) \cup \{(\perp, \perp)\})^2$ *and* $\mathrm{Init}(sk_{\mathtt{i}}, pk_{\mathtt{r}}, \mathtt{i}, ad) \rightarrow_{\$} st_{\mathtt{i}}, \mathrm{Init}(sk_{\mathtt{r}}, pk_{\mathtt{i}}, \mathtt{r}, ad) \rightarrow_{\$} st_{\mathtt{r}}$ *with* $ad \in \{0,1\}^*$, *it holds for all sequences* $((op^0, \rho^0, m^0), \ldots, (op^n, \rho^n, m^n))$ *(for all* $0 \leq l \leq n$ *with* $op^l \in \{e, d\}, \rho^l \in \{\mathtt{i}, \mathtt{r}\}, m^l \in \{0,1\}^*$*) that are executed for* $op^l = e$ *by invoking* $\mathrm{Enc}(sk_{\rho^l}, st_{\rho^l}, m^l) \rightarrow_{\$} (st_{\rho^l}, c^l, \varsigma^l)$, $MSC_\rho \leftarrow MSC_\rho \| (m^l, \varsigma^l, c^l)$, *and for* $op^l = d$ *by invoking* $(m^l_\circ, \varsigma^l, c^l) \| MSC_{\bar{\rho}} \leftarrow MSC_{\bar{\rho}}, \mathrm{Dec}(sk_{\rho^l}, st_{\rho^l}, c^l) \rightarrow_{\$} (st_{\rho^l}, m^l_*, \varsigma^l_*)$, *that if* $m^l_* \neq \perp$, *then* $m^l_* = m^l_\circ$ *and* $\varsigma^l_* = \varsigma^l$, *and that for all* $l^* < l$ *with* $op^l = op^{l^*} = e$ *and* $\rho^{l^*} = \rho^l$ *it holds that* $\varsigma^{l^*} \leq \varsigma^l$.

4.2 Execution Environment

Here we describe the execution environment for our fACCE security experiment. In our model we allow the analysis of multiple security properties, and indeed allow these properties to be reached at different points during the protocol execution. As such we follow a similar approach as the multi-stage key exchange (MSKE) model. An important difference between the models is that the MSKE model is itself

[8] One could imagine that the round-trips in the protocol may serve as stages. However, one can only define round-trips in a protocol execution if both session participants can be observed (which is not the case when considering active adversaries).

based upon standard Bellare-Rogaway key exchange frameworks, and allows the analysis of security goals for each key separately. The fACCE model does not assess the security of individual keys, but instead makes use of the stage numbers ς output to assess the security of each transmitted message. This means that one can specify for each stage which security properties need to be reached by the protocol in order to provide *security*. As a result, while one security property may not be reached in an early stage (and thus the adversary could trivially attack communication in this stage), later stages may reach this security property. Consequently, we need to separate the security experiment challenges that the adversary is to solve in each stage. We therefore define stage-specific challenge bits and freshness flags (opposed to one single challenge bit and a *static* freshness condition). The latter are dynamically checked and modified during the security game.

Further differences from the MSKE model are that we use a generic partnering notion (instead of protocol-dependent session identifiers), define authentication flexibly (e.g., unilateral authentication does not necessarily mean server authentication), provide a metric to meaningfully compare security statements of differing yet similar protocols, and due to the ACCE nature of our model, provide statements on 'internally used' symmetric keys (for which composition results of the MSKE models can naturally provide no generic guarantees).

We consider a set of n_P parties each (potentially) maintaining a long-term key pair $\{(sk_1, pk_1), \ldots, (sk_{n_P}, pk_{n_P})\}$, $(sk_i, pk_i) \in \mathcal{SK} \times \mathcal{PK}$. In addition to the key pair, a variable $corr_i \in \{0,1\}$ is stored for every party $i \in [n_P]$ by the security experiment, which indicates whether the respective secret key was given to the adversary (via $\mathsf{OCorrupt}$, see Section 4.4).

Each party can participate in up to n_S sessions. We denote both the set of variables that are specific for a session s of party i as well as the identifier of this session as π_i^s. In addition to the local variables specific to each protocol, we list the set of per-session variables that we require for our model below. In order to derive or modify a variable x of session π we write $\pi.x$ to specify this variable.

- $\rho \in \{\mathtt{i}, \mathtt{r}\}$: The role of the session in the protocol execution (i.e., initiator or responder).
- $pid \in [n_P]$: The session partner's identifier.
- ad: Data associated with this session (provided as parameter at session initialization to Init).
- $T_e[\cdot], T_d[\cdot] \in \{0,1\}^*$: Arrays of sent or received ciphertexts. After every invocation of Enc or Dec of a session π_i^s, the respective ciphertext is appended to $\pi_i^s.T_e$ or $\pi_i^s.T_d$ respectively.
- $st \in \mathcal{ST}$: All protocol-specific local variables[9].
- $rand \in \{0,1\}^*$: Any random coins sampled by this session participant.
- $(b_1, b_2, b_3, ...)$: A vector of challenge bits the adversary is to guess (one bit for each stage).
- $(fr_1, fr_2, fr_3, ...)$: A vector of freshness flags indicating whether the security of a stage in the session is considered to have been trivially broken by adversarial behavior.
- $rr \in \{0,1\}$: A flag indicating whether the session-specific $\underline{\text{r}}$andom coins have been $\underline{\text{r}}$evealed to the adversary.[10]

At the beginning of the game, for all sessions π_i^s the following initial values are set: $\pi_i^s.T_e$, $\pi_i^s.T_d$, $\leftarrow \epsilon$, $\pi_i^s.rr \leftarrow 0$, $\pi_i^s.fr_{\varsigma^*} \leftarrow 1$ for all $\varsigma^* \in \mathbb{N}$, and $\pi_i^s.rand \leftarrow_\$ \{0,1\}^*$, $\pi_i^s.b_{\varsigma^*} \leftarrow_\$ \{0,1\}$ for all $\varsigma^* \in \mathbb{N}$ are sampled.

Furthermore a set of ciphertexts $Rpl \leftarrow \emptyset$ is maintained in the security game, that are declared to initiate a non-fresh ($\underline{\text{re}}\text{p}\underline{\text{l}}$ayed) session.

Partnering In order to define security in a flexible manner, we need to define partnering for sessions in the environment. Partnering is defined over the ciphertexts provided to/by the adversary via the oracles that let sessions encrypt and decrypt ($\mathsf{OEnc}, \mathsf{ODec}$). Intuitively, a session has an honest partner if everything that the honest partner received via ODec was sent by the session via OEnc (without modification) and vice versa, and at least one of the two parties received a ciphertext at least once[11]. This definition considers the asynchronous nature of the established channel, leading to a *matching conversation*-like partnering definition for fACCE.

[9] For Noise these are (cf. Figure 2) the key(s) k or k_s, k_r, the chaining key ck, the hash value h, the ephemeral exponent, and the nonce(s) n or n_s, n_r – but only as long as they need to be stored.

[10] Please note that this variable is only necessary for the full model, described in Appendix B.

[11] Note that this definition of *honest partnering* is symmetric (i.e., if a session π_i^s has an honest partner π_j^t, then this π_j^t has π_i^s as an honest partner as well).

Definition 3 (Honest Partner). *π_j^t is an honest partner of π_i^s if $\pi_i^s.pid = j$, $\pi_j^t.pid = i$, $\pi_i^s.\rho \neq \pi_j^t.\rho$, $\pi_i^s.ad = \pi_j^t.ad$, $\pi_i^s.T_d$ is a prefix of $\pi_j^t.T_e$, and $\pi_j^t.T_d$ is a prefix of $\pi_i^s.T_e$ where at least one prefix is not empty (i.e., for $a = |\pi_j^t.T_d|$, $b = |\pi_i^s.T_d|$ such that $a > 0$ if $\pi_i^s.\rho = \mathtt{i}$ and $b > 0$ if $\pi_i^s.\rho = \mathtt{r}$ then $\forall\ 0 \leq \alpha < a : (\pi_i^s.T_e[\alpha] = \pi_j^t.T_d[\alpha])$ and $\forall\ 0 \leq \beta < b : (\pi_i^s.T_d[\beta] = \pi_j^t.T_e[\beta])$). If π_i^s already received ciphertexts from π_j^t, then π_j^t is an honest partner of π_i^s only if there exists no other honest partner π^* of π_i^s (i.e., if $b > 0$ then there is no π^* such that π^* is an honest partner of π_i^s and $\pi^* \neq \pi_j^t$).*

We provide a corresponding pseudocode definition of function Partner(i, s) that computes the honest partner(s) of a session π_i^s in Appendix D Figure 7. Please note that after encrypting without decrypting yet, the initiator may have multiple honest partners (if the resulting ciphertexts are forwarded to multiple sessions). Due to the last requirement in Definition 3, our partnering notion requires that after decrypting once, there must only exist one honest partner anymore.

In Appendix C.1 we discuss the advantages and drawbacks of our partnering notions as well as extensions to consider unreliable networks.

4.3 Flexible Security Notion

Our model enables us to analyze *levels* of authentication and confidentiality – even for different stages within one protocol execution. Our framework allows us to distinguish precisely if and when (a) Authentication and Integrity, (b) KCI resistance, (c) Forward-secrecy, (d) Resistance against randomness reveal, and (e) Resistance against replay attacks are reached.

In order to keep the presentation of the model comprehensible, we initially disregard the security properties *KCI resistance* and *resistance against randomness reveal* in the main body but describe the full model in Appendix B. Our full proofs of the Noise protocol patterns in Appendix E correspond to the full model.

As the MSKE model [16] also considers multiple security properties *stage-wise*, it defines security properties for each stage separately. This makes sense for protocols that may increase *and* decrease security levels after reaching certain security properties, but for protocols that monotonically increase security properties, a simpler notion suffices[12]. Since most real-world protocols adhere to this, our security definition is indexed by ten integers, called counters, ($\mathtt{au}^\mathtt{i}, \mathtt{au}^\mathtt{r}, \mathtt{kc}^\mathtt{i}, \mathtt{kc}^\mathtt{r}, \mathtt{fs}, \mathtt{eck}, \mathtt{rl}^\mathtt{i}, \mathtt{rl}^\mathtt{r}, \mathtt{rp}^\mathtt{i}, \mathtt{rp}^\mathtt{r}$) that indicate from which stage the respective property is achieved. Since properties can be established asymmetrically (e.g., a responder authenticates itself to an unauthenticated initiator in the first stage), some counters are indexed by role $\rho \in \{\mathtt{i}, \mathtt{r}\}$ (for initiator and responder respectively). One can think of each counter as a reference 'rung' on the 'ladder' of stages from which on the specified security property is achieved by the respectively analyzed protocol. Thus, as soon as the protocol output a certain stage that equals a counter (the protocol says, it reached the indicated 'rung' on the 'ladder'), all messages that are transmitted thereafter reach the corresponding security property. As described above, we regard and explain the counters that are necessary for the full model ($\mathtt{kc}^\mathtt{i}, \mathtt{kc}^\mathtt{r}, \mathtt{eck}, \mathtt{rl}^\mathtt{i}, \mathtt{rl}^\mathtt{r}$) only in Appendix B. We describe the (remaining) counters below:

1. $\mathtt{au}^\rho$ defines the stage required for ρ to be <u>au</u>thenticated. This means that it is hard to break the authenticity and integrity of ciphertexts from a party with role ρ (i.e., parties with role $\bar{\rho}$ reject ciphertexts if the origin is not an honest partner) if the stage number ς (output by Dec for the peer with $\bar{\rho}$) is greater or equal to $\mathtt{au}^\rho$.
2. $\mathtt{fs}$ defines the stage from which <u>f</u>orward-<u>s</u>ecrecy (with respect to both session participants' long-term secrets) is reached. It is hard, for a stage $\varsigma \geq \mathtt{fs}$, to break the confidentiality of ciphertexts, even if both parties were corrupted (*or* unless one of the session participants' random coins were revealed to the adversary; see the full model).[13]
3. $\mathtt{rp}^\rho$ defines the stage from which a fully revealed session state of ρ cannot be used to <u>rep</u>lay and reestablish the session. Our partnering definition already forbids replay attacks – except for the first ciphertext(s) from an initiator to a responder. These 'unpreventably' replayed ciphertexts establish secrets in the local session state of the sender and receivers such that these local states might be

[12] We make use of this but one can easily extend our model to allow analyses of protocols with decreasing security properties.

[13] We remark that there are more fine grained variants of forward-secrecy (such as e.g., forward-secrecy with respect to only one session participant's long-term secret) on which we comment in Appendix C.5.

dependent. The local state of a session for which stage $\varsigma \geq \mathtt{rp}^{\rho}$ was reached, hence, must not contain secrets that affect the communication's security of non-partnered sessions. Essentially this defines when the session state is independent from other non-partnered sessions such that messages, encrypted or decrypted under this state, are confidential even though other non-partnered sessions' states are obtained by the adversary.[14]

We remark that our partnering notion already excludes protocols from consideration that allow replay attacks for ciphertexts sent by an initiator that has already received a ciphertext once, and for any ciphertext sent by a responder. Thus, only replays of ciphertexts, sent by an initiator to (multiple) responder(s) without any reply from the latter, must be considered in the security experiment. These replay attacks cannot be prevented if the receiver's long-term secret is defined static (which we do in contrast to e.g., [20]) and the initiator has never received a ciphertext. Our definition of replay attack resistance consequently focuses on the security damage that is caused by such replay attacks: it considers how soon the secrets, established by a (replayed) ciphertext, are independent among the sender and the receivers of this replayed ciphertext. Hence, a session's secrets are recovered from a replay attack if they cannot be used to obtain information on other sessions' secrets.

If a property is never reached in the specified protocol, then the respective counter is set to ∞ (e.g. for protocol with unauthenticated initiators, $\mathtt{au}^{\mathtt{i}} = \infty$).

4.4 Adversarial Model

In order to model active attacks in our environment, we provide the $\mathsf{OInit}, \mathsf{OEnc}, \mathsf{ODec}$ oracles to an adversary $\mathcal{A}$, who can use them to control communication among sessions, together with the oracles $\mathsf{OCorrupt}$, $\mathsf{OReveal}$, (and in the full model $\mathsf{ORevealRandomness}$) which are all answered by the security experiment.

Since our security definition becomes simpler and more clear by considering trivial attacks during the execution of the security game (not only as a separate freshness condition evaluated after the adversary terminated), we describe the excluded trivial attacks and rewarded real attacks inline. The considered security properties are denoted as bullet point symbols below (in case they are not generically applicable).

While a fraction of previous work (especially in ACCE) defined integrity (and authentication) properties together with confidentiality goals within one game (based on the dense DAE notion of Rogaway and Shrimpton [41]), we treat these two properties similar to the original AEAD notion of Rogaway [40]: the game maintains a win flag (to indicate whether the adversary broke authenticity or integrity of ciphertexts) and embeds challenge bits in the encryption (in order to model indistinguishability of ciphertexts). In order to win the security game, adversary $\mathcal{A}$ either has to trigger $\mathsf{win} \leftarrow 1$ or output the correct challenge bit $\pi_i^s.b_\varsigma$ of a specific session stage ς at the end of the game.

– $\mathsf{OInit}(i, s, j, \rho, ad)$ initializes a session π_i^s (if not yet initialized) of party i to party j, invoking fACCE.Init$(sk_i, pk_j, \rho, ad) \rightarrow_{[\pi_i^s.rand]} \pi_i^s.st$ under $\pi_i^s.rand$. It also sets $\pi_i^s.\rho \leftarrow \rho$, $\pi_i^s.pid \leftarrow j$, and $\pi_i^s.ad \leftarrow ad$. This oracle provides no return value. All subsequent invocations of Enc, Dec of this session participant use $\pi_i^s.rand$ for obtaining randomness. Finally, the freshness flags are updated by invoking $\text{Fresh}_{\mathtt{fs}}()$ (see Figure 3).
– $\mathsf{OEnc}(i, s, m_0, m_1)$ triggers the encryption of message m_b for $b = \pi_i^s.b_\varsigma$ by invoking Enc$(sk_i, \pi_i^s.st, m_b)$ $\rightarrow_{[\pi_i^s.rand]} (st', c, \varsigma)$ for an initialized π_i^s if $|m_0| = |m_1|$ and for $\varsigma = 0$ (i.e., confidentiality is not yet achieved) it must hold that $m_0 = m_1$. It updates the session specific variables $\pi_i^s.st \leftarrow st'$, returns (c, ς) to the adversary, and appends c to $\pi_i^s.T_e$ if encryption succeeds.
– $\mathsf{ODec}(i, s, c)$ triggers invocation of Dec$(sk_i, \pi_i^s.st, c) \rightarrow_{[\pi_i^s.rand]} (st', m, \varsigma)$ for an initialized π_i^s and returns (m, ς) if π_i^s has no honest partner, or returns ς otherwise (since challenges from the encryption oracle would otherwise be trivially leaked). Finally c is appended to $\pi_i^s.T_d$ if decryption succeeds.
  Excluding trivial attacks:
  $\mathtt{fs}$: Since decryption can change the honesty of partners, the freshness flags are updated regarding corruptions by invoking $\text{Fresh}_{\mathtt{fs}}()$ (see Figure 3).

[14] Please note that this does not *only* cover replayability of payloads but also, how long secrets in a session state can be leaked to allow replaying *and* comprehending another session. One could extend the precision regarding replayability by further considering properties such as post-compromise security, whether the long-term key was already used, or whether asymmetric cryptography is used (which we consider not useful in this setting).

au: The consideration of trivial attacks regarding authentication are a combination of the stage at which the protocol reaches authentication and corruptions of the participants' long-term secrets. If the received ciphertext was not sent by a session of the intended partner (i.e., there exists no honest partner) and

1. party i is corrupted (i.e., $corr_i = 1$), then all following stages are marked un-fresh ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^*$), since this is a KCI attack.[15]
2. neither party i nor the session's intended partner are corrupted (i.e., $corr_i = corr_{\pi_i^s.pid} = 0$) and authentication of the partner was not reached yet (i.e., $\varsigma < \mathtt{au}^{\pi_i^s.\bar{\rho}}$), then all following stages are marked un-fresh until authentication will be reached ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^* < \mathtt{au}^{\pi_i^s.\bar{\rho}}$), since this is a (temporary) trivial impersonation of the partner towards π_i^s.[16]
3. only the session's intended partner is corrupted (i.e., $corr_{\pi_i^s.pid} = 1 \neq corr_i$) and authentication of the partner was not reached yet or is reached with this received ciphertext (i.e., $\varsigma \leq \mathtt{au}^{\pi_i^s.\bar{\rho}}$), then all following stages are marked un-fresh ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^*$), since this is (and will continue to be) a trivial impersonation of the partner towards π_i^s.

Rewarding real attacks:

au: Similarly to detecting trivial attacks, real attacks are rewarded by considering when authentication is reached in the respective protocol execution and if the participants' long-term secrets are corrupted.
The adversary breaks authentication (and thereby $\mathsf{win} \leftarrow 1$ is set) if the received ciphertext was not sent by a session of the intended partner but was successfully decrypted (i.e., there exists no honest partner and the output state is $st' \neq \perp$), the stage is still fresh ($\pi_i^s.fr_\varsigma = 1$), and

1. this is the first authenticated ciphertext ($\varsigma = \mathtt{au}^{\pi_i^s.\bar{\rho}}$), and neither party i nor the intended partner is corrupted ($corr_i = corr_{\pi_i^s.pid} = 0$), or
2. this is a later authenticated ciphertext ($\varsigma > \mathtt{au}^{\pi_i^s.\bar{\rho}}$) and party i is not corrupted ($corr_i = 0$) as this would otherwise be a KCI attack.

– $\mathsf{OCorrupt}(i) \rightarrow sk_i$ outputs the long-term secret key sk_i of party i, sets $corr_i \leftarrow 1$, and updates the freshness flags by invoking $\mathrm{Fresh}_{\mathtt{fs}}()$.
– $\mathsf{OReveal}(i, s) \rightarrow \pi_i^s.st$ outputs the current session state $\pi_i^s.st$.

Excluding trivial attacks:

- Revealing the session-state trivially leaks sufficient information to solve embedded challenge bits[17]. Hence $\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ is set for all stages ς^*.
- Similarly sufficient information is leaked to solve challenge bits embedded in ciphertexts to and from *all* honest partners π_j^t (and to impersonate π_i^s towards them). As such, $\pi_j^t.fr_{\varsigma^*} \leftarrow 0$ is set for all stages ς^* of these honest partners.

rp: In case the revealed secrets enable the adversary to obtain secrets of non-partnered sessions due to a replay attack ($\varsigma < \mathtt{rp}^{\pi_i^s.\rho}$ where ς was output by π_i^s's last OEnc or ODec query) then the first ciphertext in this session is declared to induce non-fresh sessions via $Rpl \leftarrow Rpl \cup \{c\}$ where $c \leftarrow \pi_i^s.T_e[0]$ if $\pi_i^s.\rho = \mathtt{i}$ or $c \leftarrow \pi_i^s.T_d[0]$ if $\pi_i^s.\rho = \mathtt{r}$ (such that all sessions starting with this ciphertext are also marked non-fresh)[18].

Freshness regarding Corruptions The definition of forward-secrecy, based on counter $\mathtt{fs}$, is straight forward: if either the own long-term secrets or the intended partner's long-term secrets were corrupted (i.e., $corr_i = 1 \vee corr_{\pi_i^s.pid} = 1$), then only stages that provide forward-secrecy are marked fresh for the respective session (i.e., $\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma^* < \mathtt{fs}$). We formally define this property via function

[15] Please note that in the reduced model (in contrast to the full model in Appendix B), resistance against KCI attacks is not required.

[16] If the partner authenticates later, then the protocol must ensure that this early trivial impersonation is detected. Consequently, this attack is not treated trivial anymore after the partner's authentication.

[17] Since we do not consider forward-secrecy within sessions, the secret session state is considered to harm security of the whole session lifetime independent of when the state is revealed.

[18] One can easily define this trivial attack more specifically depending on whether this first ciphertext is authenticated and/or designated to a certain party. Depending on that, the secrets established by this ciphertext would only be valid among specific session (cf. [22]). For clarity and simplicity, we generically treat the ciphertext replayable solely. Please note that a state, revealed before the first ciphertext was sent/received (i.e., $c = \epsilon$), should not harm security of other sessions.

$\mathrm{Fresh}_{\mathtt{fs}}()$ (see Figure 3) because this simplifies the definition of freshness in the full model (via $\mathrm{Fresh}_{full}()$) in which also sessions' random coins can be revealed.

```
$\mathrm{Fresh}_{\mathtt{fs}}()$:
For all $i \in [n_P]$, for all $s \in [n_S]$:
    $\mathtt{ctr} \leftarrow \min(\varsigma^* : \pi_i^s.fr_{\varsigma^*} = 1)$
    If $corr_i = 1 \vee corr_{\pi_i^s.pid} = 1$:
        $\mathtt{ctr} \leftarrow \max(\mathtt{ctr}, \mathtt{fs})$
    $\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma^* < \mathtt{ctr}$
```

Fig. 3: Function for updating freshness flags after each oracle invocation, only considering long-term secrets' corruption. $\mathrm{Fresh}_{full}()$ that defines freshness also under the reveal of random coins can be found in Figure 5.

4.5 Security Definition

The notion of fACCE security is captured as a game played by an adversary $\mathcal{A}$ in which the sessions are implemented as described above. At the beginning of the game, n_P long-term key pairs (pk_i, sk_i) $\forall i \in [n_P]$ are generated via fACCE.KGen and the respective public keys are provided to $\mathcal{A}$ as a parameter on the invocation (i.e., the start of the game). $\mathcal{A}$ interacts with the game via the queries described above and eventually terminates, potentially outputting a tuple (i, s, ς, b').

We can now turn to defining the security of a fACCE protocol. Straightforwardly, a fACCE protocol is secure if it is correct and the advantage of any PPT algorithm $\mathcal{A}$ in guessing a challenge bit b for a fresh stage or breaking authentication or integrity is negligible.

Definition 4 (Flexible ACCE Security). *We say that an adversary $\mathcal{A}$ breaks a flexible ACCE protocol* fACCE *with authentication stages $(\mathtt{au}^{\mathtt{i}}, \mathtt{au}^{\mathtt{r}})$, forward-secrecy stage $\mathtt{fs}$, and replayability resistance stages $(\mathtt{rp}^{\mathtt{i}}, \mathtt{rp}^{\mathtt{r}})$, when $\mathcal{A}$ terminates and outputs (i, s, ς, b'), if there either exists a session π_i^s such that $\pi_i^s.b_\varsigma = b'$, $\pi_i^s.fr_\varsigma = 1$, and $\pi_i^s.T_e[0] \notin Rpl$ for $\pi_i^s.\rho = \mathtt{i}$, or $\pi_i^s.T_d[0] \notin Rpl$ for $\pi_i^s.\rho = \mathtt{r}$ respectively, or* $\mathsf{win} = 1$. *We define the advantage of an adversary $\mathcal{A}$ breaking a flexible ACCE protocol* fACCE *as* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{fACCE}} = (2 \cdot \Pr[\pi_i^s.b_\varsigma = b'] - 1) + \Pr[\mathsf{win} = 1]$.

4.6 Mapping Stages to Round-Trips

The indication of stages by the protocol (and the respective related security properties) does not provide meaningful information for a classification of protocols. For example, it might be tempting to compare protocols and only consider those that provide a confidential channel within one round-trip. As a consequence, we map stages to round-trips in which they are reached (in an honest execution). This does not only help for comparison, but also helps to highlight impossibilities or tradeoffs (such as reaching authentication quickly while, against an active attacker, identity hiding is preserved). The number of round-trips, in which stages are reached, are actually separated into half-round-trips (i.e. a goal can be reached with RT $0.5, 1, 1.5, 2, \ldots$ or is implicitly reached, indicated by 0).

Example The Noise pattern `XK` depicted in Figure 2 has four stages (one for each handshake ciphertext). The properties reached with the last handshake ciphertext equal for messages transmitted later in the protocol. In this pattern, the responder is authenticated towards the initiator in round-trip 1 (with ciphertext g^b, c_1), while authentication in the other direction is reached in round-trip 1.5 when c_2, c_3 are transmitted. We denote this by $\mathrm{RT}(\mathtt{au}^{\mathtt{r}}) = 1$ and $\mathrm{RT}(\mathtt{au}^{\mathtt{i}}) = 1.5$ (with $\mathtt{au}^{\mathtt{r}} = 2, \mathtt{au}^{\mathtt{i}} = 3$). Since the ephemeral DH values are mixed into the key material within the first round-trip (i.e., with handshake ciphertexts g^a, c_0 and g^b, c_1), forward-secrecy is reached for this and all subsequent round-trips. Hence we have $\mathrm{RT}(\mathtt{fs}) = 1, \mathtt{fs} = 2$.

4.7 On the Necessity of one Holistic Model

Our definition of flexible ACCE considers multiple security properties simultaneously (as opposed to having separate definitions for each regarded security property). While the reduced model, presented above, is still similarly comprehensible (or complex) as usual key exchange models, the full version, presented in Appendix B, requires more effort to be understood. In order to reduce complexity, it could seem useful to regard the security properties independently and then assemble the results. In the following we explain why this approach produces more complexity, less comprehensibility, and is partially impossible.

Certainly confidentiality and authenticity themselves are not immediately dependent – and actually our model treats successful adversaries differently, depending on which of both properties has been broken. However, if protocols are considered that are not immediately resistant against replay attacks (and hence this is an own security property), or if the adversary is allowed to reveal sessions' random coins (and hence the resistance against these attacks is also an own security property), then these considerations affect both, confidentiality and authenticity. These additional considerations are particularly further dimensions of confidentiality and authenticity rather than distinct security properties (and thus cannot be analyzed independently). For example, KCI resistance is not an orthogonal security property but only a dimension of authenticity. Similarly, resistance against replay attacks refers to a dimension of authenticity (that crucially also affects confidentiality). As these *dimensions* of confidentiality and authenticity additionally affect each other, a security definition that captures only a subset of them cannot be assembled with another disjoint security definition subset. Finally, as for example trivial attacks against authenticity invalidate confidentiality challenges, it is reasonable to consider both properties within one definition.

We describe an example protocol in Appendix C.3 Figure 6 that (without a proof) provides authentication under secure randomness (i.e., $\mathtt{au}^{\mathtt{i}} = x < \infty$, $x \le \mathtt{eck} = y \le \infty$) and confidentiality under randomness reveal (i.e., $\mathtt{eck} = z < x < \infty$, $\mathtt{au}^{\mathtt{i}} = \infty$), but not both properties simultaneously (i.e., there is an attack for defining $\mathtt{au}^{\mathtt{i}} = x$ and $\mathtt{eck} = z$ in the same analysis). This depicts that in a model with a reduced subset of properties (as in our reduced model that forbids randomness reveal) one can derive different results than in the full model (in which randomness can be revealed). We emphasize that the counters in our theorems all hold for the full model, even though we provide proof sketches with respect to the reduced model.

Intuitively, the example protocol from Appendix C.3 Figure 6 requires for authentication fresh randomness that can only be revealed after the respective party is authenticated. Confidentiality can, however, be achieved even if the randomness is revealed at the protocol start. Due to revealing the randomness early, an adversary can impersonate the authenticating party during the protocol run. Thus, independent definitions and analyses can result in meaningless security statements.

5 Protocol Analyses

In this section, we provide an overview of our results of analyzing the Noise Protocol framework in our new fACCE model. Our main contribution is the full proofs of Noise Patterns `N`, `NN`, `NX`, `NK`, and `X`, `XN`, `XX`, `XK`. To demonstrate the robustness of our security model and how the proof structure mirrors the construction of the Noise Patterns, we give a detailed look of the proofs of Noise Patterns `N`, and `NK` in the reduced model here. We extend these proofs, considering further security properties in the full model, together with the proofs for the remaining mentioned patterns in Appendix E.

We chose Noise Patterns `N` and `NK` because (due to their simplicity) they comprehensibly provide an idea of the general proof structure and they show how Noise patterns can be built upon another. As the handshake of `NK` extends `N`'s handshake by a half round-trip, the former also achieves further security properties.

Generic Proof Structure The modular design of the Noise Protocol Framework allow us to write proofs that have a reasonably generic structure. While the proof for each specific Noise Pattern is distinct, each proof is, on a high level, split into two cases:

- The adversary has forged a ciphertext successfully, and sent it to a session that does *not* detect the forgery (or abort the protocol run). This case may be further split into multiple cases depending on which ciphertext in the Noise Pattern the adversary has managed to forge.
- The adversary has guessed the challenge bit correctly when it terminates the experiment.

As specific freshness conditions need to hold for each case, as otherwise forging a ciphertext or guessing the challenge bit is trivial, we determine which $\mathsf{OCorrupt}/\mathsf{OReveal}$ (and in the full model $\mathsf{ORevealRandomness}$) queries cannot have been issued. Thus each case has a distinct pair of $\mathsf{OCorrupt}/\mathsf{OReveal}$ queries that have not been issued to the session π_i^s and its partner session (where π_i^s has either accepted the forged ciphertext, or the adversary has output (i, s, ς, b')). In both cases we use a tailored $\mathsf{PRF\text{-}ODH}$ assumption, depending on which pair of queries (targeting long-term DH shares, state secrets, or, in the full model, ephemeral DH shares that depend on random coins) have *not* been issued, to replace the appropriate Diffie-Hellman public values and shared Diffie-Hellman secrets (using the ODH oracles to compute any additional secrets using the DH secret keys, if necessary). Afterwards, we iteratively replace intermediate secrets derived during the protocol execution using PRF assumptions on the underlying key derivation function. Finally, we use a single (or potentially series of) AEAD assumptions to replace the encryptions of ciphertexts sent to, and decryption of ciphertexts arriving at, the session π_i^s. Any adversary capable of distinguishing these changes is able to break one of the underlying assumptions used, and depending on which case we are in, either:

– The adversary is unable to forge a ciphertext to the session π_i^s, or
– The adversary is unable to guess the challenge bit b with non-negligible probability.

This (high-level) description effectively captures the strategy we use to prove our statements about the Noise Patterns that we analyze.

Mapping Noise's Security Statements to our Model's Counters First we introduce the stage counters $(\mathtt{au}^{\mathtt{i}}, \mathtt{au}^{\mathtt{r}}, \mathtt{fs}, \mathtt{rp}^{\mathtt{i}}, \mathtt{rp}^{\mathtt{r}})$, $(\mathtt{kc}^{\mathtt{i}}, \mathtt{kc}^{\mathtt{r}}, \mathtt{eck}, \mathtt{rl}^{\mathtt{i}}, \mathtt{rl}^{\mathtt{r}})$[19], used in our theorems of each proof, that define the exact modeled security. We also explain how they relate to the round-trips in the protocol execution of the respective Noise pattern. For each of the base patterns of the Noise specification, the stage at which the respective security property is reached is listed in Table 1. As stage numbers ς, output by the Enc, Dec algorithms, are defined as integers, we assume the Noise patterns to output a counter as stage number with every algorithm invocation, starting by 1 and always incremented by 1 until no further security properties are reached.[20] As a result, also in presence of an adversary, the stage numbers during the handshake match twice the round-trip in which they are output by Enc or Dec respectively.

The counters/round-trips for authentication and KCI resistance $(\mathtt{au}^{\rho}, \mathtt{kc}^{\rho})$ are directly lifted from the Noise specification [38]. As the definition of the remaining security properties deviate from the specification (or are not specified there at all), these are the first round-trips and stages that achieve the respective goals.

While, regarding forward-secrecy, the Noise specification differentiates among role dependent weak and strong variants of long-term secrets' corruptions, our consideration of forward-secrecy focuses on the relation between corruptions of long-term secrets and the reveal of sessions' random coins. Consequently, the counter $\mathtt{fs}$ is only partially derived from the Noise specification.

Resistance against replay attacks in the Noise specification only considers the adversary's ability to successfully let multiple sessions receive the same sent ciphertext. However, local state variables (like an ephemeral symmetric encryption key or a DH exponent), established by a ciphertext, can be exploited by an adversary to attack other sessions that sent or received the same (replayed) ciphertext. Such state variables may stay in the local state even after the replay attack "is over" (i.e., after only a unique honest partner exist). As the adversary is allowed to the reveal the local state, our definition of replay attack resistance goes beyond others in the literature (e.g., [17]) and the Noise specification: it says that resistance against replay attacks is reached if the local state of a session is independent of any other session's state (except from the respective unique honest partner).

5.1 Proof Sketch of Noise Pattern N

Theorem 1. *The Noise protocol* $\mathtt{N}$ *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (\infty, \infty)$*, forward-secrecy* $\mathtt{fs} = \infty$*, and replay resistance* $\mathtt{rp} = (\infty, \infty)$ *where* $\varsigma = 1$ *is output with the*

[19] The latter are only relevant for the proofs in the full model.

[20] Thus the first encryption of the initiator and the first decryption of the responder output 1, the responder's first encryption and the initiator's first decryption output 2, and so on. Note that, during the handshake, Noise patterns run in a strict alternating form.

	$\mathtt{au}^{\mathtt{i}}$	$\mathtt{au}^{\mathtt{r}}$	$\mathtt{fs}$	$\mathtt{rp}^{\mathtt{i}}$	$\mathtt{rp}^{\mathtt{r}}$	$\mathtt{kc}^{\mathtt{i}}$	$\mathtt{kc}^{\mathtt{r}}$	$\mathtt{eck}$	$\mathtt{rl}^{\mathtt{i}}$	$\mathtt{rl}^{\mathtt{r}}$
$\mathtt{N}^*$	∞	∞	∞	∞	∞	∞	∞	∞	1	∞
$\mathtt{X}^*$	1	∞	∞	∞	∞	∞	∞	1	1	∞
$\mathtt{K}$	1	∞	∞	∞	∞	∞	∞	1	1	∞
$\mathtt{NN}^*$	∞	∞	2	2	0	∞	∞	∞	∞	∞
$\mathtt{NK}^*$	∞	2	2	2	2	∞	2	∞	1	∞
$\mathtt{NX}^*$	∞	2	2	2	0	∞	2	∞	2	∞
$\mathtt{XN}^*$	3	∞	2	2	0	3	∞	∞	∞	3
$\mathtt{XK}^*$	3	2	2	2	2	3	2	∞	1	3
$\mathtt{XX}^*$	3	2	2	2	0	3	2	∞	2	3
$\mathtt{KN}$	3	∞	2	2	0	3	∞	∞	∞	2
$\mathtt{KK}$	1	2	2	2	2	3	2	1	1	2
$\mathtt{KX}$	3	2	2	2	0	3	2	∞	2	2
$\mathtt{IN}$	3	∞	2	2	0	3	∞	∞	∞	2
$\mathtt{IK}$	1	2	2	2	2	3	2	1	1	2
$\mathtt{IX}$	3	2	2	2	0	3	2	∞	2	2

Table 1: Stages at which the respective security properties are reached. In an honest execution, stage x is reached (and thus returned by the protocol via output ς) at round-trip $\mathrm{RT}(x) = x/2$.The right half of columns depicts the counters for security properties that are only considered in the full model. $\mathtt{au}^\rho, \mathtt{kc}^\rho$ were extracted from Noise's specification [38]; $\mathtt{fs}, \mathtt{rp}^\rho$ are related to their definition in the specification (but adapted to our model). $\mathtt{rl}^\rho, \mathtt{eck}$ were defined purely with respect to the model. We give proofs for the patterns marked with a *.

first ciphertext and $\varsigma > 1$ for all remaining in an honest execution. That is, for an adversary $\mathcal{A}$ against the flexible ACCE security game (defined in section 4) one can efficiently define adversaries $\mathcal{B}_{\mathsf{coll}}$ against the collision resistance of H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the PRF-ODH assumption* ms-PRF-ODH *with respect to group G and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{N},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

Proof (Sketch).
We give below a sketch of the proof of Noise Pattern $\mathtt{N}$. For the full details of the proof, see Appendix E.1. First, recall that the adversary breaks authentication (and thus, sets $\mathsf{win} \leftarrow 1$) if the received ciphertext was not sent by a session owned by the intended partner, but was successfully decrypted. Since in $\mathtt{N}$, only the initiator sends encrypted ciphertexts, we can restrict our focus to a responder party that successfully decrypts ciphertexts from a non-honest partner. However, by the security experiment, it is required that if $\mathsf{win} \leftarrow 1$ in the responder session in any given stage, the stage must still be fresh. Since the initiator never authenticates (i.e., $\mathtt{au}^{\mathtt{i}} = \infty$), $\mathcal{A}$ injecting the ciphertext from a non-honest partner would cause $\pi_i^s.fr_\varsigma \leftarrow 0 \;\forall\, \varsigma$, and thus $\Pr[\mathsf{win} \leftarrow 1] = 0$. Now we can turn our focus to an adversary attempting to guess the challenge bit b for an initiator session (since the responder session never encrypts in Noise Pattern $\mathtt{N}$, and thus the behavior of a responder session is independent of its challenge bits). If $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b'), where $\pi_i^s.\rho = \mathtt{i}$, and $\pi_i^s.fr_1 = 1$ then $\mathcal{A}$ cannot issue a $\mathsf{OCorrupt}(\pi_i^s.pid)$ query as $\mathtt{fs} = \infty$, and $\mathtt{au}^{\mathtt{r}} = \infty$. We show that under such restrictions, $\mathcal{A}$ has a negligible advantage in guessing a challenge bit b for the session π_i^s.

We begin with the standard fACCE experiment defined in Section 4. In **Game 1**, the challenger aborts on hash collisions, and thus this modification is reduced to the collision resistance of H.

In the next two games (**Game 2**, **Game 3**) we guess the index (i, s) of the session π_i^s, as well as the index j of the honest partner π_j^t and abort if either $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma, b')$ such that $(i^*, s^*) \neq (i, s)$, or if $\mathcal{A}$ initialises π_i^s such that $\pi_i^s.pid \neq j$. In what follows, the challenger playing the fACCE game "knows" at the beginning of the experiment the index of the session that $\mathcal{A}$ will target, and its intended partner j.

In **Game 4** the challenger replaces the value of ck, k_0 with uniformly random values $\widetilde{ck}, \widetilde{k_0}$ in the test session π_i^s and its honest partner (if one exists). This change is reduced to the PRF-ODH assumption under the ephemeral DH share of π_i^s and the long-term DH share of j, since neither its state can be revealed, nor j can be corrupted without marking the stages un-fresh. Since $\mathtt{rp} = \infty$, however, the reduction also has to simulate sessions $\pi_j^{t^*}$ where g^a, c_0 (the ciphertext sent by π_i^s) is replayed to $\pi_j^{t^*}$. The reduction simply sets the value of ck, k_0 in such sessions to $\widetilde{ck}, \widetilde{k_0}$ to ensure consistency.

In **Game 5** the challenger replaces the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ in π_i^s and its honest partner (if one exists). This replaces the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$ via the PRF assumption of KDF since by **Game 4**, $\widetilde{ck}$ is already uniformly random and independent of the protocol flow.

In **Game 6** the challenger flips a bit $\bar{b}$ and encrypts $m_{\bar{b}}$ in the first invocation of OEnc by π_i^s, instead of $m_{\pi_i^s.b_1}$. This change is reduced to the AEAD security of the encryption under the key $\widetilde{k_0}$. Since $\widetilde{k_0}$ is uniformly random and independent (by **Game 4**), this change is sound.

In **Game 7**, (and similarly to **Game 6**) the challenger flips again a bit $\bar{b}$ and encrypts $m_{\bar{b}}$ in the all remaining invocations of OEnc by π_i^s instead of m_b where b is $\pi_i^s.b_{\varsigma^*}$ for all $\varsigma^* > 1$. This change is reduced to the AEAD security of the encryption under the key $\widetilde{k_{\mathtt{i}}}$. Since $\widetilde{k_{\mathtt{i}}}$ is uniformly random and independent (by **Game 5**), this change is sound.

In **Game 7**, the behavior of π_i^s is independent of the test bits $\pi_i^s.b_1$, $\pi_i^s.b_2$ thus $\mathcal{A}$ has no advantage in guessing either, and thus summing our probabilities we find our result.

5.2 Proof Sketch of Noise Pattern NK

Theorem 2. *The Noise protocol* NK *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (\infty, 2)$*, forward-secrecy* $\mathtt{fs} = 2$*, and replay resistance* $\mathtt{rp} = (2, 2)$*. That is, for an algorithm* $\mathcal{A}$ *against the flexible ACCE security game (defined in section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the PRF-ODH assumptions* $\mathsf{ms\text{-}PRF\text{-}ODH}$ *and* $\mathsf{nn\text{-}PRF\text{-}ODH}$ *with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE}}_{\mathrm{NK},n_P,n_S,\mathcal{A}} \leq 2 \cdot \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} &+ n_P^2 n_S \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}} \\ &+ 2 \cdot \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{G,p,\mathrm{KDF},\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} \Big) + \ n_P^2 n_S^2 \Big(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} \\ &+ \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}} \Big)\end{aligned}$$

Proof (Sketch).
We split the analysis into three cases:

1. **Case A**: $\mathcal{A}$ causes to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 (i.e when π_i^s with $\pi_i^s.\rho = \mathtt{i}$ outputs $\varsigma = 2$ during a decryption call).
2. **Case B**: $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the first ciphertext g^a, c_0 (i.e when π_i^s with $\pi_i^s.\rho = \mathtt{r}$ outputs $\varsigma = 1$ during a decryption call).
3. **Case C**: $\mathcal{A}$ outputs (i, s, ς', b') but does not cause π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 (if $\pi_i^s.\rho = \mathtt{i}$) or when π_i^s processes the ciphertext g^a, c_0 (if $\pi_i^s.\rho = \mathtt{r}$).

We begin by treating **Case A**. We know in this case that $\mathcal{A}$ cannot have issued a $\mathsf{OCorrupt}(\pi_i^s.pid)$ query because breaking authentication of a corrupted peer is a trivial attack (as $\mathsf{au}^{\mathtt{r}} = 2$). In order to set $\mathsf{win} \leftarrow 1$ for π_i^s as defined above, the value of $\pi_i^s.fr_2$ must, however, be 1.

The first three games (**Game** $\mathbf{1}, \mathbf{2}, \mathbf{3}$) proceed identically to the proof of N. That is, we abort when a hash-collision is detected, and guess the index (i, s) of the session π_i^s, as well as the index j of the intended partner.

In **Game 4**, we introduce an abort event that occurs if $\mathsf{win} \leftarrow 1$ in the test session. Afterwards, in **Game 5**, the challenger replaces the value of ck, k_0 with uniformly random values $\widetilde{ck}, \widetilde{k_0}$ in the test session π_i^s and its honest partner (if one exists). This change is reduced to the $\mathsf{PRF\text{-}ODH}$ assumption under the ephemeral DH share of π_i^s and the long-term DH share of j, since neither its state can be revealed, nor j can be corrupted without marking the stages un-fresh. The reduction also must simulate sessions $\pi_j^{t^*}$ where g^a, c_0 (the ciphertext sent by π_i^s) is replayed to $\pi_j^{t^*}$. It therefore sets the computation of ck, k_0 in such sessions to $\widetilde{ck}, \widetilde{k_0}$ to ensure consistency.

In **Game 6** the challenger replaces the concretely computed values ck, k_1 in π_i^s and its honest partner (if one exists), with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. As by **Game 4** the input $\widetilde{ck}$ is already chosen uniformly random, distinguishing this game from the previous one can be reduced to the PRF security of the KDF.

In **Game 7**, the challenger flips a bit $\bar{b}$ and encrypts $m_{\bar{b}}$ in the first invocation of OEnc by π_i^s instead of m_b where b is $\pi_i^s.b_1$. This change is reduced to the AEAD security of the encryption under the key $\widetilde{k_0}$.

Game 8 proceeds identically to **Game 7**, except that it responds to OEnc or ODec queries directed to π_i^s or π_j^t outputting $\varsigma = 2$ from $\mathcal{A}$ (i.e. when using the key $\widetilde{k_1}$) as opposed to using the key $\widetilde{k_0}$ and it aborts if π_i^s receives g^b, c_1 but has no honest partner. This is reduced to the AEAD security of the AEAD scheme.

Note that by **Game 8** we abort if no honest session owned by j has output g^b, c_1. Thus in **Game 8**, $\mathcal{A}$ has no advantage in triggering the event $abort_{win}$ due to π_i^s processing g^b, c_1.

We now treat **Case B** and note $\mathcal{A}$ cannot cause π_i^s such that $\pi_i^s.\rho = \mathtt{r}$ and $\pi_i^s.fr_1 = 1$ to set $\mathsf{win} \leftarrow 1$, as $\mathsf{au}^{\mathtt{i}} = \infty$. We can now treat **Case C**.

We follow standard procedure and define an abort event to trigger when we find1 a hash-collision, guess the index (i, s) of the session π_i^s, and the index (j, t) of the honest partner π_j^t. By **Case A** and **Case B**, there *must* exist such an honest partner for the beginning of stage $\varsigma = 3$.

In what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, except for a change in notation.

At this point, we need to split the analysis into two sub-cases:

1. **Case C.1**: $\mathcal{A}$ has not issued a $\mathsf{OCorrupt}(j)$ query.
2. **Case C.2**: $\mathcal{A}$ has potentially issued a $\mathsf{OCorrupt}(j)$ query.

In **Case C.1 Game 4**, we replace ck, k_0 with uniformly random values $\widetilde{ck}, \widetilde{k_0}$ in π_i^s and its honest partner which can be reduced to the $\mathsf{PRF\text{-}ODH}$ assumption (as in **Case A, Game 4**).

In **Game 5** and **Game 6**, we replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$, and subsequently replace $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$ via the PRF assumption on KDF.

Game 7 and **Game 8** proceed identically to **Case C Game 6** but flip and use independent challenge bits when answering queries to OEnc if key $\widetilde{k_0}$ is used in stage $\varsigma = 1$ (**Game 7**) and when keys $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$ are used in stage $\varsigma = 3$. Furthermore **Game 8** aborts if π_i^s processes a ciphertext in $\varsigma = 3$ but there exists no honest partner. The modifications in both games are reduced to the AEAD security. In **Case C.1, Game 8**, the behavior of π_i^s is independent of the test bits $\pi_i^s.b_1$, $\pi_i^s.b_3$ and $\mathcal{A}$ has no advantage in setting $\mathsf{win} \leftarrow 1$ for the initiator.

We now treat **Case C.2**, in which we know that $\mathcal{A}$ potentially has issued a $\mathsf{OCorrupt}(j)$ query. Since $\mathtt{fs} = 2$, by Table 3 any adversary that issues a $\mathsf{OCorrupt}(j)$ sets $\pi_i^s.fr_1 \leftarrow 0$ and outputting $(i, s, 1, b')$ will lose $\mathcal{A}$ the game. As a result, in **Case C.2** we cannot prove the security of payload data sent in the first ciphertext.

In **Game 4**, we replace the computation of ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$ in π_i^s and its honest partner, which is reduced to the $\mathsf{PRF\text{-}ODH}$ assumption based on the ephemeral DH shares of both sessions. We follow in **Game 5** by replacing $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$ via a PRF assumption on KDF.

Game 6 proceeds similarly to **Case C.1 Game 7** by encrypting $m_{\bar{b}}$ for a randomly flipped bit $\bar{b}$ when using the key $\widetilde{k_1}$. Finally, **Game 7** proceeds identically to **Case C.1 Game 8** by encrypting $m_{\bar{b}'}$ for another randomly flipped bit $\bar{b}'$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$). Furthermore **Game 7** also aborts if π_i^s processes a ciphertext in $\varsigma = 3$ but there exists no honest partner. The modifications in both games are reduced to the AEAD security.

In **Case C.2, Game 7**, the behavior of π_i^s is independent of the test bits $\pi_i^s.b_2$, $\pi_i^s.b_3$ as well as setting $\mathsf{win} \leftarrow 1$ when π_i^s processes a ciphertext is aborted, and thus $\mathcal{A}$ has no advantage in winning the game. It is possible for $\mathcal{A}$ to later issue a $\mathsf{OCorrupt}(j)$ query, in this case $\pi_i^s.fr_1 \leftarrow 0, \pi_j^t.fr_1 \leftarrow 0$, and $\mathcal{A}$ cannot win by outputting the tuple $(i, s, 1, b')$.

6 Discussion

In our work, we introduced a flexible authenticated and confidential channel establishment (fACCE) security model that extracts the core idea of an ACCE protocol, and generalizes to capture a range of fine-grained security guarantees. We demonstrate the robustness of the novel fACCE model by considering the Noise Protocol framework, and proving the security of eight separate Noise Patterns, each with distinct security goals and guarantees in the face of varying threat models.

Many Noise patterns attempt some form of identity hiding, by using early established keys to encrypt and exchange long-term public keys, the examination of which we leave for future work. Similarly, Noise includes many *pattern modifiers* that extend the 15 base Noise patterns, for instance, the use of preshared keys, which we also leave for future work.

It is important to note that the aim of our model is explicitly not to propose the next *super-strong* notion of security (since all security properties except confidentiality can be analyzed optionally but not all independently), but to propose a generic model- and proof-approach.

As the main reason for basing a protocol analysis on an ACCE model is the design of the specific analyzed protocol (i.e., the use of keys during the key establishment), it is surprising that all previous ACCE model definitions were heavily influenced by the concept of composing a channel establishment protocol from a key exchange protocol and a channel protocol. Consequently, our results systematize and contribute to the understanding of the generic – composition-independent – primitive *confidential and authenticated channel establishment* since we provide the first generic syntax definition for this primitive and illustrate how to flexibly define security for it.

Bibliography

[1] Aviram, N., Schinzel, S., Somorovsky, J., Heninger, N., Dankel, M., Steube, J., Valenta, L., Adrian, D., Halderman, J.A., Dukhovni, V., Käsper, E., Cohney, S., Engels, S., Paar, C., Shavitt, Y.: DROWN: breaking TLS using sslv2. In: USENIX Security (2016)

[2] Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO (1993)

[3] Bergsma, F., Dowling, B., Kohlar, F., Schwenk, J., Stebila, D.: Multi-ciphersuite security of the secure shell (SSH) protocol. In: CCS (2014)

[4] Blazy, O., Bossuat, A., Bultel, X., Fouque, P.A., Onete, C., Pagnin, E.: Said: Reshaping signal into an identity-based asynchronous messaging protocol with authenticated ratcheting. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2019)

[5] Brendel, J., Fischlin, M., Günther, F., Janson, C.: PRF-ODH: relations, instantiations, and impossibility results. In: CRYPTO (2017)

[6] Brzuska, C., Jacobsen, H.: A modular security analysis of EAP and IEEE 802.11. In: PKC (2017)

[7] Brzuska, C., Jacobsen, H., Stebila, D.: Safely exporting keys from secure channels - on the security of EAP-TLS and TLS key exporters. In: EUROCRYPT (2016)

[8] Brzuska, C., Smart, N.P., Warinschi, B., Watson, G.J.: An analysis of the EMV channel establishment protocol. In: CCS (2013)

[9] Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: CRYPTO (2003)

[10] Checkoway, S., Niederhagen, R., Everspaugh, A., Green, M., Lange, T., Ristenpart, T., Bernstein, D.J., Maskiewicz, J., Shacham, H., Fredrikson, M.: On the practical exploitability of dual EC in TLS implementations. In: USENIX Security (2014)

[11] Chen, S., Jero, S., Jagielski, M., Boldyreva, A., Nita-Rotaru, C.: Secure communication channel establishment: Tls 1.3 (over tcp fast open) vs. quic. Cryptology ePrint Archive, Report 2019/433 (2019), `https://eprint.iacr.org/2019/433`

[12] Cohn-Gordon, K., Cremers, C.J.F., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. In: IEEE EuroS&P (2017)

[13] Donenfeld, J.A.: Wireguard: Next generation kernel network tunnel. In: NDSS (2017)

[14] Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: CCS (2015)

[15] Dowling, B., Paterson, K.G.: A cryptographic analysis of the wireguard protocol. In: ACNS (2017)

[16] Fischlin, M., Günther, F.: Multi-stage key exchange and the case of google's QUIC protocol. In: CCS (2014)

[17] Fischlin, M., Günther, F.: Replay attacks on zero round-trip time: The case of the TLS 1.3 handshake candidates. In: IEEE EuroS&P (2017)

[18] Giesen, F., Kohlar, F., Stebila, D.: On the security of TLS renegotiation. In: CCS (2013)

[19] Green, M.: The strange story of "extended random" (2017), `https://blog.cryptographyengineering.com/2017/12/19/the-strange-story-of-extended-random/`

[20] Günther, F., Hale, B., Jager, T., Lauer, S.: 0-rtt key exchange with full forward secrecy. In: EUROCRYPT (2017)

[21] Günther, F., Mazaheri, S.: A formal treatment of multi-key channels. In: CRYPTO (2017)

[22] Hale, B., Jager, T., Lauer, S., Schwenk, J.: Simple security definitions for and constructions of 0-rtt key exchange. In: ACNS (2017)
[23] Inc., W.: Whatsapp encryption overview (2016), `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`, technical white paper
[24] Jaeger, J., Stepanovs, I.: Optimal channel security against fine-grained state compromise: The safety of messaging. In: CRYPTO (2018)
[25] Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of tls-dhe in the standard model. In: CRYPTO (2012)
[26] Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: Authenticated confidential channel establishment and the security of TLS-DHE. J. Cryptology 30(4) (2017)
[27] Kobeissi, N.: Noise explorer (2018), `https://noiseexplorer.com/`
[28] Kobeissi, N., Nicolas, G., Bhargavan, K.: Noise explorer: Fully automated modeling and verification for arbitrary noise protocols. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2019)
[29] Kohlar, F., Schäge, S., Schwenk, J.: On the security of tls-dh and tls-rsa in the standard model. Cryptology ePrint Archive (2013), `https://eprint.iacr.org/2013/367`
[30] Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: CRYPTO (2013)
[31] LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger security of authenticated key exchange. In: ProvSec (2007)
[32] Li, Y., Schäge, S.: No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In: CCS (2017)
[33] Li, Y., Schäge, S., Yang, Z., Kohlar, F., Schwenk, J.: On the security of the pre-shared key ciphersuites of TLS. In: PKC (2014)
[34] Lipp, B., Blanchet, B., Bhargavan, K.: A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2019)
[35] Lychev, R., Jero, S., Boldyreva, A., Nita-Rotaru, C.: How secure and quick is quic? provable security and performance analyses. In: IEEE S&P (2015)
[36] Morrissey, P., Smart, N.P., Warinschi, B.: A modular security analysis of the TLS handshake protocol. In: ASIACRYPT (2008)
[37] Möller, B., Duong, T., Kotowicz, K.: This poodle bites: Exploiting the ssl 3.0 fallback (2014), `https://www.openssl.org/~bodo/ssl-poodle.pdf`
[38] Perrin, T.: The noise protocol framework (2017), `http://noiseprotocol.org/noise.html`, revision 33
[39] Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: CRYPTO (2018)
[40] Rogaway, P.: Authenticated-encryption with associated-data. In: CCS (2002)
[41] Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: EUROCRYPT (2006)
[42] Rösler, P., Mainka, C., Schwenk, J.: More is less: On the end-to-end security of group chats in signal, whatsapp, and threema. In: IEEE EuroS&P (2018)

A Further Security Assumptions

A.1 Collision-Resistant Hash Functions

Definition 5 (Collision-resistant hash function). *A* collision-resistant hash function *is a deterministic algorithm* $\mathrm{H} : \{0,1\}^* \to \{0,1\}^\lambda$ *which, given a bit string* m *outputs a hash value* $w \leftarrow \mathrm{H}(m)$ *in the hash space* $\{0,1\}^\lambda$*. We define the advantage of an adversary* $\mathcal{A}$ *breaking the collision-resistance of the hash function* H *is* $\mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{A}} = \Pr\left[(m, m') \leftarrow \mathcal{A} : \mathrm{H}(m) = \mathrm{H}(m'), m \neq m'\right]$.

A.2 Pseudo-Random Function Security

Definition 6 (PRF Security). *A pseudo-random function family is a collection of deterministic functions* $\mathsf{PRF} = \{\mathsf{PRF} : \mathcal{K} \times \mathcal{M} \to \mathcal{Z}\}$*. Given a key* k *in the keyspace* $\mathcal{K}$ *and a bit string* $m \in \mathcal{M}$*,* PRF *outputs a value* y *in the output space* $\mathcal{Z}$*. We define the security of a pseudo-random function family in the following game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*:*

1. $\mathcal{C}$ *samples a key* $k \leftarrow_\$ \mathcal{K}$ *and a bit* b *uniformly at random.*
2. $\mathcal{A}$ *can now query* $\mathcal{C}$ *with polynomially-many distinct* m_i *values, and receives either the output* $y_i \leftarrow \mathsf{PRF}(k, m_i)$ *(when* $b = 0$*) or* $y_i \leftarrow_\$ \mathcal{Z}$ *(when* $b = 1$*).*
3. $\mathcal{A}$ *terminates and outputs a bit* b'*.*

We say that $\mathcal{A}$ *wins the* PRF *security game if* $b' = b$ *and define the advantage of an adversary* $\mathcal{A}$ *in breaking the* pseudo-random function security *of a PRF family* PRF *as* $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}} = |2 \cdot \Pr[b' = b] - 1|$.

Intuitively, a PRF is secure if for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}$ is negligible.

A.3 AEAD security

In the following, we will provide a definition of authenticated encryption with associated data as given in [40].

Definition 7 (AEAD scheme). *We define an* authenticated encryption scheme with associated data *(AEAD) as a tuple of algorithms* $\mathsf{AEAD} = (\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ *defined over a key space* $\mathcal{K}$*, nonces* $\mathcal{N}$*, messages* $\mathcal{M}$ *and additional data* $\mathcal{AD}$*. The encryption algorithm* Enc *is a deterministic algorithm that takes strings* $k \in \mathcal{K}$*,* $n \in \mathcal{N}$ *,* $ad \in \mathcal{AD}$ *and* $m \in \mathcal{M}$ *and returns a string* $c = \mathrm{Enc}(k, n, ad, m)$*. The decryption algorithm* Dec *is a deterministic algorithm that takes strings* $k \in \mathcal{K}$*,* $n \in \mathcal{N}$*,* $c \in \{0,1\}^*$ *and* $ad \in \mathcal{AD}$ *and returns a string* $m = \mathrm{Dec}(k, n, ad, c)$ *or* $\perp$*. We require that* $\mathrm{Dec}(k, n, ad, \mathrm{Enc}(k, n, ad, m)) = m$ *for all* $k \in \mathcal{K}$*,* $n \in \mathcal{N}$*,* $ad \in \mathcal{AD}$ *and* $m \in \mathcal{M}$*.*

In the following, we will provide an all-in-one definition of the security for authenticated encryption with associated data.

Definition 8 (AEAD security). *Let* $\mathsf{AEAD} = (\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ *be an AEAD scheme. Let* INIT0 *be the security experiment (see Figure 4) with* b *set to* 0 *and* INIT1 *be the security experiment with* b *set to* 1*. We say that the adversary wins the* AEAD *game if* $b' = b$ *and define the advantage function*

$$\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}} := |\Pr[1 \leftarrow \mathcal{A} | b = 1] - \Pr[1 \leftarrow \mathcal{A} | b = 0]|.$$

Intuitively AEAD is secure, if for all PPT algorithms $\mathcal{A}$ it holds that $\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}}$ is negligible.

A.4 The PRF-Oracle-Diffie-Hellman Assumption

In this section we introduce the definition of the PRF-ODH assumption.

Definition 9 (Generic PRF-ODH Assumption). *Let* G *be a cyclic group of order* q *with generator* g*. Let* $\mathsf{PRF} : G \times \mathcal{M} \to \mathcal{K}$ *be a function from a pseudo-random function family that takes a group element* $k \in G$ *and a salt value* $m \in \mathcal{M}$ *as input, and outputs a value* $y \in \mathcal{K}$*. We define a security notion,* lr-PRF-ODH *security, which is parameterised by:* $\mathsf{l}, \mathsf{r} \in \{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$ *indicating how often the adversary is allowed to query "left" and "right" oracles (*ODHu *and* ODHv*), where* n *indicates that no query is allowed,* s *that a single query is allowed, and* m *that multiple queries are allowed to the respective oracle. Consider the following security game* $\mathcal{G}^{\mathsf{lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}}$ *between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*.*

```
Init(1^λ)                          Enc(n, ad, m_0, m_1)
$k \leftarrow_\$ \mathrm{KGen}$                  $c_0 = \mathrm{Enc}(k, n, ad, m_0)$
$b \leftarrow_\$ \{0,1\}$                  $c_1 = \mathrm{Enc}(k, n, ad, m_1)$
$\mathcal{S} \leftarrow \{\}$                      if $(c_0 = \perp) \vee (c_1 = \perp)$:
$b' \leftarrow_\$ \mathcal{A}^{\mathsf{Enc}(),\mathsf{Dec}()}$             return $\perp$
return $(b = b')$                  else
                                     $\mathcal{S} \leftarrow \mathcal{S} \cup \{(n, ad, c_b)\}$
Dec(n, ad, c)                        return $c_b$
if $(b = 0)$:
  return $\perp$
if $(\mathrm{Dec}(k, n, ad, c) \neq \perp)$
      $\wedge((n, ad, c) \notin \mathcal{S})$:
  return $m$
else return $\perp$
```

Fig. 4: Security experiment for AEAD schemes $\mathsf{AEAD} = (\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$.

1. *The challenger $\mathcal{C}$ samples $u \leftarrow_\$ \mathbb{Z}_p$ and provides g, g^u to the adversary $\mathcal{A}$.*
2. *If $\mathsf{l} = \mathsf{m}$, $\mathcal{A}$ can issue arbitrarily many queries to oracle* ODHu*, as follows:*
   - ODHu*: on a query of the form (S, x), the challenger first checks if $S \notin G$ and returns $\perp$ if this is the case. Otherwise, it computes $y \leftarrow \mathsf{PRF}_\lambda(S^u, x)$ and returns y.*
3. *Eventually, $\mathcal{A}$ issues a challenge query x^*. The challenger $\mathcal{C}$ samples $v \leftarrow_\$ \mathbb{Z}_p$ and it is required that, for all queries (S, x) to* ODHu *made previously, if $S = g^v$, then $x \neq x^*$. This is to prevent trivial wins by $\mathcal{A}$. $\mathcal{C}$ samples a bit $b \leftarrow_\$ \{0,1\}$ uniformly at random, computes $y_0 = \mathsf{PRF}_\lambda(g^{uv}, x^*)$, and samples $y_1 \leftarrow_\$ \{0,1\}^\lambda$ uniformly at random. The challenger returns y_b to $\mathcal{A}$.*
4. *Next, $\mathcal{A}$ may issue (arbitrarily interleaved) queries to oracles* ODHu *and* ODHv*. These are handled as follows:*
   - ODHu*: on a query of the form (S, x), the challenger first checks if $S \notin G$ or if $(S, x) = (g^v, x^*)$ and returns $\perp$ if either holds. Otherwise, it returns $y \leftarrow \mathsf{PRF}_\lambda(S^u, x)$.*
   - ODHv*: on a query of the form (T, x), the challenger first checks if $T \notin G$ or if $(T, x) = (g^u, x^*)$ and returns $\perp$ if either holds. Otherwise, it returns $y \leftarrow \mathsf{PRF}_\lambda(T^v, x)$.*
5. *At some point, $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$.*

We say that the adversary wins $\mathcal{G}^{\mathsf{sym\text{-}lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}}$ if $b' = b$ and define the advantage function

$$\mathsf{Adv}^{\mathsf{lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}} = |2 \cdot \Pr[b' = b] - 1|.$$

Intuitively, the lr-PRF-ODH assumption holds if the advantage $\mathsf{Adv}^{\mathsf{lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}}$ of any PPT adversary $\mathcal{A}$ is negligible.

A.5 Symmetric Generic PRF-Oracle-Diffie-Hellman Assumption

Here we give the symmetric variant of the generic PRF-ODH assumption, introduced by Dowling and Paterson [15] to analyze the WireGuard Protocol. Since WireGuard is built upon the Noise Protocol Framework (and in particular, the pre-shared key variant of Noise Pattern `XX`, the inclusion of the assumption in our proofs is natural.

Definition 10 (Symmetric generic PRF-ODH Assumption). *Let G be a cyclic group of order q with generator g. Let $\mathsf{PRF} : G \times \mathcal{M} \to \mathcal{K}$ be a function from a pseudo-random function family that takes a group element $k \in G$ and a salt value $m \in \mathcal{M}$ as input, and outputs a value $y \in \mathcal{K}$. We define a security notion,* sym-lr-PRF-ODH *security, which is parameterised by: $\mathsf{l}, \mathsf{r} \in \{\mathsf{n}, \mathsf{s}, \mathsf{m}\}$ indicating how often the adversary is allowed to query "left" and "right" oracles (*ODHu *and* ODHv*), where* n *indicates that no query is allowed,* s *that a single query is allowed, and* m *that multiple queries are allowed to the respective oracle. Consider the following security game $\mathcal{G}^{\mathsf{sym\text{-}lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}}$ between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.*

1. *The challenger $\mathcal{C}$ samples $u, v \leftarrow_\$ \mathbb{Z}_p$ and provides g, g^u, g^v to the adversary $\mathcal{A}$.*
2. *If $\mathsf{l} = \mathsf{m}$, $\mathcal{A}$ can issue arbitrarily many queries to oracle* ODHu*, and if* $\mathsf{r} = \mathsf{m}$ and $\mathsf{sym} = \mathsf{Y}$ *to the oracle* ODHv*. These are implemented as follows:*

- ODHu*: on a query of the form* (S, x)*, the challenger first checks if* $S \notin G$ *and returns* $\perp$ *if this is the case. Otherwise, it computes* $y \leftarrow \mathsf{PRF}_\lambda(S^u, x)$ *and returns* y*.*
- ODHv*: on a query of the form* (T, x)*, the challenger first checks if* $T \notin G$ *and returns* $\perp$ *if this is the case. Otherwise, it computes* $y \leftarrow \mathsf{PRF}_\lambda(T^v, x)$ *and returns* y*.*

3. *Eventually,* $\mathcal{A}$ *issues a challenge query* x^**. It is required that, for all queries* (S, x) *to* ODHu *made previously, if* $S = g^v$*, then* $x \neq x^*$*. Likewise, it is required that, for all queries* (T, x) *to* ODHv *made previously, if* $T = g^u$*, then* $x \neq x^*$*. This is to prevent trivial wins by* $\mathcal{A}$*.* $\mathcal{C}$ *samples a bit* $b \leftarrow_\$ \{0,1\}$ *uniformly at random, computes* $y_0 = \mathsf{PRF}_\lambda(g^{uv}, x^*)$*, and samples* $y_1 \leftarrow_\$ \{0,1\}^\lambda$ *uniformly at random. The challenger returns* y_b *to* $\mathcal{A}$*.*
4. *Next,* $\mathcal{A}$ *may issue (arbitrarily interleaved) queries to oracles* ODHu *and* ODHv*. These are handled as follows:*
   - ODHu*: on a query of the form* (S, x)*, the challenger first checks if* $S \notin G$ *or if* $(S, x) = (g^v, x^*)$ *and returns* $\perp$ *if either holds. Otherwise, it returns* $y \leftarrow \mathsf{PRF}_\lambda(S^u, x)$*.*
   - ODHv*: on a query of the form* (T, x)*, the challenger first checks if* $T \notin G$ *or if* $(T, x) = (g^u, x^*)$ *and returns* $\perp$ *if either holds. Otherwise, it returns* $y \leftarrow \mathsf{PRF}_\lambda(T^v, x)$*.*
5. *At some point,* $\mathcal{A}$ *outputs a guess bit* $b' \in \{0,1\}$*.*

We say that the adversary wins $\mathcal{G}^{\mathsf{sym\text{-}lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}}$ *if* $b' = b$ *and define the advantage function*

$$\mathsf{Adv}^{\mathsf{sym\text{-}lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}} = |2 \cdot \Pr[b' = b] - 1|.$$

Intuitively, the $\mathsf{sym\text{-}lr\text{-}PRF\text{-}ODH}$ assumption holds if the advantage $\mathsf{Adv}^{\mathsf{sym\text{-}lr\text{-}PRF\text{-}ODH}}_{\mathsf{PRF},G,p,\mathcal{A}}$ of any PPT adversary $\mathcal{A}$ is negligible.

B Model with KCI Resistance and Randomness Reveal

The full model, that we use to proof security of the Noise patterns, additionally considers KCI resistance (indicated by counters $\mathtt{kc^i}, \mathtt{kc^r}$) and resistance against randomness reveal (indicated by counters $\mathtt{eck}, \mathtt{rl^i}, \mathtt{rl^r}$). We describe these counters below:

1. $\mathtt{kc}^\rho$ defines the stage required for KCI resistance of ρ to be reached. It is hard, for a stage $\varsigma \geq \mathtt{kc}^\rho$, to break authenticity of ciphertexts to a party with ρ even if it was corrupted.

As defined in the main body, for forward-secrecy it is required to be hard, for a stage $\varsigma \geq \mathtt{fs}$, to break the confidentiality of ciphertexts, unless one of the session participants' random coins were revealed to the adversary (*or* even if both parties were corrupted).
In combination with considering reveals of a session's random coins, we further introduce the following two kinds of stage counters (that supplement each other in a disjunction):

2. $\mathtt{eck}$ defines the stage from which payloads are confidential unless one of the session participants was corrupted (*or* even if *both* participants' random coins were revealed; cf. eCK model [31])[21].
3. $\mathtt{rl}^\rho$ defines the stage from which payloads are confidential unless the random coins of participant with role ρ were revealed or the long-term secret of the session partner (with role $\bar{\rho}$) was corrupted (*or* even if the random coins of $\bar{\rho}$ were revealed and ρ was corrupted).

The distinction between these four counters ($\mathtt{fs}, \mathtt{eck}, \mathtt{rl}^\rho$) makes sense when considering that the session secret can be computed based on pairwise combinations among ephemeral secrets and long-term secrets of the two session participants.

[21] While we agree with common critique of the eCK model that it does not depict insecure randomness generators, it is evident that there are threats that are depicted by a revelation of the random coins used in a session. An example would be the *extended random* extension implemented in some TLS libraries (cf. [10, 19]) or a random oracle, used to derive random coins on a low entropy input (such as the current time).

Additional Partner Notion The reveal of random coins of a session (that is only considered in this full model) does not only affect current honest partners (see Definition 3) but also sessions that previously were honest partners of the session for which the randomness was revealed. Thus we must define *Previous Honest Partner* for the full model below:

Definition 11 (Previous Honest Partner). *We say that π_j^t is a previous honest partner of π_i^s if $\pi_i^s.pid = j$, $\pi_j^t.pid = i$, $\pi_i^s.\rho = \pi_j^t.\bar{\rho}$, $\pi_i^s.ad = \pi_j^t.ad$, $\pi_i^s.T_d$ and $\pi_j^t.T_e$ have a common prefix, and $\pi_j^t.T_d$ and $\pi_i^s.T_e$ have a common prefix where at least one prefix is not empty (i.e., for $a \le |\pi_j^t.T_d|$, $b \le |\pi_i^s.T_d|$ such that $a > 0$ if $\pi_i^s.\rho = \mathtt{i}$ and $b > 0$ if $\pi_i^s.\rho = \mathtt{r}$ then $\forall\ 0 \le \alpha < a : (\pi_i^s.T_e[\alpha] = \pi_j^t.T_d[\alpha]) \wedge \forall 0\ \le \beta < b : (\pi_i^s.T_d[\beta] = \pi_j^t.T_e[\beta])$).*

We provide a pseudocode definition of function $\mathrm{PrevPartner}(i, s)$ that computes the previous honest partner(s) of a session π_i^s in Appendix D Figure 7. The main differences towards *honest partner* are that: (a) In *previous honest partners* a and b can be less than or equal $|\pi_j^t.T_d|$ and $|\pi_i^s.T_d|$ respectively (meaning that π_i^s and π_j^t were honest partners once) and due to this; (b) It is not (and actually cannot be) required that there exists only one *previous honest partner*.

B.1 Adversarial Model

In our full model, the adversary has – in addition to calling $\mathsf{OInit}, \mathsf{OEnc}, \mathsf{ODec}, \mathsf{OCorrupt}, \mathsf{OReveal}$ – the ability to reveal sessions' random coins via the oracle $\mathsf{ORevealRandomness}$. Below we describe the oracles that the adversary can call in the full model, including the security experiment's internal treatment of trivial and real attacks. The only difference in the oracles $\mathsf{OInit}, \mathsf{OEnc}, \mathsf{OCorrupt}, \mathsf{OReveal}$ of the full model towards the reduced model in the main body (see Section 4) is that $\mathrm{Fresh}_{full}()$ is invoked to update the freshness flags instead of $\mathrm{Fresh}_{\mathtt{fs}}()$. The computation of the freshness flag via $\mathrm{Fresh}_{full}()$ is described below in Subsection B.2, Table 3, and Figure 5. Consequently we do not describe these negligibly modified oracles in full details here again.

– $\mathsf{ODec}(i, s, c)$ triggers invocation of $\mathrm{Dec}(sk_i, \pi_i^s.st, c) \rightarrow_{[\pi_i^s.rand]} (st', m, \varsigma)$ for an initialized π_i^s and returns (m, ς) if π_i^s has no honest partner (since challenges from the encryption oracle would otherwise be trivially leaked), or returns ς otherwise. Finally c is appended to $\pi_i^s.T_d$ if decryption succeeds.
Excluding trivial attacks: In Table 2 we provide an overview over the trivial and successful attacks with respect to authentication and KCI resistance.

$\mathtt{fs}, \mathtt{eck}, \mathtt{rl}^{\rho}$: Since decryption can change the honesty of partners, the freshness flags are updated regarding corruptions and the reveal of random coins by invoking $\mathrm{Fresh}_{full}()$ (see Table 3 and Figure 5).

$\mathtt{au}, \mathtt{kc}$: To exclude trivial attacks regarding authentication and KCI resistance, it is important to note that attacks against the latter only make sense when considering the former as the main goal. Trivial attacks are then a combination of reached goals by the protocol and corruptions of the participants' long-term secrets.
If the received ciphertext was not sent by a session of the intended partner (i.e., there exists no honest partner) and authentication of the partner

1. was not reached yet (i.e., $\varsigma < \mathtt{au}^{\pi_i^s.\bar{\rho}}$), then all following stages are marked un-fresh until authentication will be reached ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \le \varsigma^* < \mathtt{au}^{\pi_i^s.\bar{\rho}}$), since this is a (temporarily) trivial impersonation of the partner towards π_i^s.
2. is reached with this ciphertext (i.e., $\varsigma = \mathtt{au}^{\pi_i^s.\bar{\rho}}$), and
   (a) intended partner $\pi_i^s.pid$ is corrupted ($corr_{\pi_i^s.pid} = 1$), then all following stages are marked un-fresh ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \le \varsigma^*$), since this is a trivial impersonation of the partner towards π_i^s.
   (b) intended partner $\pi_i^s.pid$ is not corrupted, party i is corrupted, and KCI resistance is not reached yet (i.e., $\varsigma < \mathtt{kc}^{\pi_i^s.\rho}$, $corr_{\pi_i^s.pid} \ne corr_i = 1$), then all following stages are marked un-fresh until KCI resistance would be reached ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \le \varsigma^* < \mathtt{kc}^{\pi_i^s.\rho}$), since this is a (temporarily) trivial *key compromise impersonation* of the partner towards π_i^s.[22]
3. was reached before, but KCI resistance is not reached yet, and party i is corrupted (i.e., $\mathtt{au}^{\pi_i^s.\bar{\rho}} < \varsigma < \mathtt{kc}^{\pi_i^s.\rho}$, $corr_i = 1$), then all following stages are marked un-fresh until KCI resistance would be reached ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \le \varsigma^* < \mathtt{kc}^{\pi_i^s.\rho}$), since this is a (temporarily) trivial *key compromise impersonation* of the partner towards π_i^s.[22]

[22] Obviously when KCI resistance is reached, key compromise impersonation is no longer considered trivial.

4. is reached or was reached before, and the session's random coins were revealed (i.e., $\mathtt{au}^{\pi_i^s.\bar{\rho}} \leq \varsigma$, $\pi_i^s.rr = 1$), then all following stages are marked un-fresh ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^*$), since this is a trivial impersonation of the partner towards π_i^s.[23]

$\mathtt{kc}$ The game requires a session to defend a trivial KCI attack as soon as KCI resistance is reached. If, after trivially breaking authenticity by conducting a KCI attack, the adversary corrupts the partner $\pi_i^s.pid$ to continue this trivial impersonation, this necessarily needs to be marked as such. Hence, if the received ciphertext was not sent by a session of the intended partner (i.e., there exists no honest partner), KCI resistance is reached with this ciphertext ($\varsigma = \mathtt{kc}^{\pi_i^s.\rho}$), and the intended partner is corrupted ($corr_{\pi_i^s.pid} = 1$), then all following stages are marked un-fresh ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^*$), since this is a trivial impersonation of the partner towards π_i^s.

Rewarding real attacks:

$\mathtt{au}, \mathtt{kc}$: Similarly to detecting trivial attacks, real attacks are rewarded by considering the goals that are defined to be reached by the protocol and the corruptions of the participants' long term secrets. The adversary breaks authentication (and thereby $\mathsf{win} \leftarrow 1$ is set) if the received ciphertext was not sent by a session of the intended partner but was successfully decrypted (i.e., there exists no honest partner and the output state is $st' \neq \bot$), the stage is still fresh ($\pi_i^s.fr_\varsigma = 1$), π_i^s's randomness was not revealed ($\pi_i^s.rr = 0$)[23], and

1. this is the first authenticated ciphertext ($\varsigma = \mathtt{au}^{\pi_i^s.\bar{\rho}}$), the intended partner $\pi_i^s.pid$ is not yet corrupted, and, if KCI resistance was not reached yet, then party i is not yet corrupted ($corr_{\pi_i^s.pid} = 0$ and if $\varsigma < \mathtt{kc}^{\pi_i^s.\rho}$ then $corr_i = 0$), or
2. this is a later authenticated ciphertext ($\varsigma > \mathtt{au}^{\pi_i^s.\bar{\rho}}$) and, if KCI resistance was not reached yet, then party i is not yet corrupted (if $\varsigma < \mathtt{kc}^{\pi_i^s.\rho}$ then $corr_i = 0$).

– $\mathsf{ORevealRandomness}(i, s) \to rand$ outputs the randomness $\pi_i^s.rand$ sampled by party i in its session π_i^s. The session is marked $\pi_i^s.rr \leftarrow 1$ and the freshness flags are updated by invoking $\mathrm{Fresh}_{full}()$.

	$\varsigma < \mathtt{au}^{\pi_i^s.\bar{\rho}}$	$\varsigma = \mathtt{au}^{\pi_i^s.\bar{\rho}}$ $\varsigma < \mathtt{kc}^{\pi_i^s.\rho}$	$\varsigma = \mathtt{au}^{\pi_i^s.\bar{\rho}}$ $\varsigma = \mathtt{kc}^{\pi_i^s.\rho}$	$\varsigma > \mathtt{au}^{\pi_i^s.\bar{\rho}}$ $\varsigma < \mathtt{kc}^{\pi_i^s.\rho}$	$\varsigma > \mathtt{au}^{\pi_i^s.\bar{\rho}}$ $\varsigma = \mathtt{kc}^{\pi_i^s.\rho}$	$\varsigma > \mathtt{kc}^{\pi_i^s.\rho}$
$corr_i = corr_{pid} = 0$	$\mathtt{au}^{\pi_i^s.\bar{\rho}}$	win	win	win	win	win
$corr_i = 0 \wedge corr_{pid} = 1$	∞	∞	∞	win	win	win
$corr_i = 1 \wedge corr_{pid} = 0$	$\mathtt{kc}^{\pi_i^s.\rho}$	$\mathtt{kc}^{\pi_i^s.\rho}$	win	$\mathtt{kc}^{\pi_i^s.\rho}$	win	win
$corr_i = corr_{pid} = 1$	∞	∞	∞	∞	∞	win

Table 2: Definition of trivial and successful attacks regarding authentication and KCI resistance based on the corruption of the receiver and its intended partner on the receipt of a forged ciphertext. For trivial attacks the value in the cell $\mathtt{ctr}$ denotes the first stage for which the freshness flag is not erased: $\forall \varsigma \leq \varsigma^* < \mathtt{ctr} : \pi_i^s.fr_{\varsigma^*} \leftarrow 0$. A successful attack is denoted by cells containing win, meaning that if $\pi_i^s.fr_\varsigma = 1$ and the state st output by π_i^s's invocation of Dec is not $\bot$, then $\mathsf{win} \leftarrow 1$ is set. Note that always $\mathtt{au}^{\bar{\rho}} \leq \mathtt{kc}^{\rho}$.
As long as KCI resistance is not yet reached, the corrupted sender's long-term secret can help the adversary to impersonate the sender towards the receiver (and thereby a trivial impersonation can still be conducted). Only if authentication with KCI resistance is reached, the sender's long-term secret can be corrupted without affecting security.
Please note that the counters in the second last row denote a shortcut compared to the textual definition of the trivial attacks.

B.2 Freshness regarding Corruption and Reveal of Randomness

Based on the definition of the counters $\mathtt{fs}, \mathtt{eck}, \mathtt{rl^i}$, and $\mathtt{rl^r}$, the effects of corruptions and the revelation of sessions' random coins can be mapped onto the freshness flag $\pi_i^s.fr_\varsigma$ of all sessions π_i^s (note that this is stage dependent). To determine the freshness of a session π_i^s, the corruption of the session owner's long-term secrets (i.e. $corr_i$), the intended partner's long-term secrets (i.e., $corr_{\pi_i^s.pid}$), the revelation of the sessions' own random coins (i.e., $\pi_i^s.rr$), and the revelation of honest partners' random coins (i.e., $\pi_j^t.rr$; $j = \pi_i^s.pid$ by definition) need to be considered. Not only do the current honest partners affect the

[23] We discuss on this trivial attack and its necessity, in Appendix C.4. One can think of it as a KCI attack based on the receiver's ephemeral secrets (instead of its long-term secrets).

Freshness of π_i^s based on:		$corr_{\pi_i^s.pid} = 1$	$\pi_j^t.rr = 1$	$\pi_j^t.rr = 1$ $\wedge corr_{\pi_i^s.pid} = 1$
		$\min(\mathtt{rl}^{\pi_i^s.\bar{\rho}}, \mathtt{fs})$	$\min(\mathtt{rl}^{\pi_i^s.\rho}, \mathtt{eck})$	∞
$corr_i = 1$	$\min(\mathtt{rl}^{\pi_i^s.\rho}, \mathtt{fs})$	$\mathtt{fs}$	$\mathtt{rl}^{\pi_i^s.\rho}$	∞
$\pi_i^s.rr = 1$	$\min(\mathtt{rl}^{\pi_i^s.\bar{\rho}}, \mathtt{eck})$	$\mathtt{rl}^{\pi_i^s.\bar{\rho}}$	$\mathtt{eck}$	∞
$\pi_i^s.rr = 1$ $\wedge corr_i = 1$	∞	∞	∞	∞

Table 3: Freshness of session π_i^s is defined over corruptions of both parties $i, \pi_i^s.pid$ and randomness reveals of the session and its (previous) honest partner(s) π_j^t denoted in the first row and column. The respective cells denote from which stage on security is reached under these conditions.

security of a session, but also sessions that were honest partners previously. In Table 3, the stage from which on security is reached is defined based on the corruptions and revelations of random coins of both session participants. The cells immediately result from the definition of the counters $\mathtt{fs}, \mathtt{eck}, \mathtt{rl}^{\rho}$. The oracle description above updates the freshness flags based on this table via function Fresh$_{full}$() (defined in Figure 5).

```
Fresh$_{full}$():
For all $i \in [n_P]$, for all $s \in [n_S]$:
    $\mathtt{ctr} \leftarrow \min(\varsigma^* : \pi_i^s.fr_{\varsigma^*} = 1)$
    If $corr_{\pi_i^s.pid} = 1$:
        $\mathtt{ctr} \leftarrow_{\max} \min(\mathtt{rl}^{\pi_i^s.\bar{\rho}}, \mathtt{fs})$
    If $\exists (j,t) \in \text{PrevPartner}(i,s) : \pi_j^t.rr = 1$:
        $\mathtt{ctr} \leftarrow_{\max} \min(\mathtt{rl}^{\pi_i^s.\rho}, \mathtt{eck})$
    If $(corr_{\pi_i^s.pid} = 1$ and $\exists (j,t) \in \text{PrevPartner}(i,s) :$
      $\pi_j^t.rr = 1)$ or $(corr_{\pi_i^s.pid} = 1$ and $\pi_i^s.rr = 1)$:
        $\mathtt{ctr} \leftarrow \infty$
    If $corr_i = 1$:
        $\mathtt{ctr} \leftarrow_{\max} \min(\mathtt{rl}^{\pi_i^s.\rho}, \mathtt{fs})$
        If $corr_{\pi_i^s.pid} = 1$:
            $\mathtt{ctr} \leftarrow_{\max} \mathtt{fs}$
        If $\exists (j,t) \in \text{PrevPartner}(i,s) : \pi_j^t.rr = 1$:
            $\mathtt{ctr} \leftarrow_{\max} \mathtt{rl}^{\pi_i^s.\rho}$
    If $\pi_i^s.rr = 1$:
        $\mathtt{ctr} \leftarrow_{\max} \min(\mathtt{rl}^{\pi_i^s.\bar{\rho}}, \mathtt{eck})$
        If $corr_{\pi_i^s.pid} = 1$:
            $\mathtt{ctr} \leftarrow_{\max} \mathtt{rl}^{\pi_i^s.\bar{\rho}}$
        If $\exists (j,t) \in \text{PrevPartner}(i,s) : \pi_j^t.rr = 1$:
            $\mathtt{ctr} \leftarrow_{\max} \mathtt{eck}$
    $\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma^* < \mathtt{ctr}$
```

Fig. 5: Function for updating freshness flags after each oracle invocation based on Table 3 in the full model. $x \leftarrow_{\max} y$ is a shortcut notion for $x \leftarrow \max(x, y)$. For the definition of function PrevPartner(i, s) see Figure 7.

In Appendix C.5 we provide a discussion on the meaning of revealed random coins and consider reasonable critique of the eCK model. We therefore propose an alternative to revealing random coins and describe how our model can be adapted respectively (which results in an even stronger adversary).

B.3 Security Definition

The security definition of fACCE in the full model equals Definition 4 except that also counters $(\mathtt{kc}^{\mathtt{i}}, \mathtt{kc}^{\mathtt{r}}, \mathtt{eck}, \mathtt{rl}^{\mathtt{i}}, \mathtt{rl}^{\mathtt{r}})$ are considered:

Definition 12 (Flexible ACCE Security). *We say that an adversary $\mathcal{A}$ breaks a flexible ACCE protocol* fACCE *with authentication stages* $(\mathtt{au}^{\mathtt{i}}, \mathtt{au}^{\mathtt{r}})$*, key compromise impersonation resilience stages* $(\mathtt{kc}^{\mathtt{i}}, \mathtt{kc}^{\mathtt{r}})$*, forward-secrecy stage* $\mathtt{fs}$*, eCK security stage* $\mathtt{eck}$*, stages for confidentiality based on long-term secrets and random coins respectively* $(\mathtt{rl}^{\mathtt{i}}, \mathtt{rl}^{\mathtt{r}})$*, and replayability resistance stages* $(\mathtt{rp}^{\mathtt{i}}, \mathtt{rp}^{\mathtt{r}})$*, when* $\mathcal{A}$ *terminates and outputs* (i, s, ς, b')*, there either exists a session* π_i^s *such that* $\pi_i^s.b_\varsigma = b'$*,* $\pi_i^s.fr_\varsigma = 1$*, and* $\pi_i^s.T_e[0] \notin Rpl$ *for* $\pi_i^s.\rho = \mathtt{i}$ *or* $\pi_i^s.T_d[0] \notin Rpl$ *for* $\pi_i^s.\rho = \mathtt{r}$ *respectively, or* $\mathsf{win} = 1$*. We define the advantage of an adversary* $\mathcal{A}$ *breaking a flexible ACCE protocol* fACCE *as* $\mathsf{Adv}_{\mathcal{A}}^{\text{fACCE}} = (2 \cdot \Pr[\pi_i^s.b_\varsigma = b'] - 1) + \Pr[\mathsf{win} = 1]$*.*

Intuitively, a flexible ACCE protocol fACCE is secure if it is correct and $\mathsf{Adv}_{\mathcal{A}}^{\text{fACCE}}$ is negligible for all probabilistic algorithms $\mathcal{A}$ running in polynomial-time.

C Further Discussion of our Model

The security definition of channel establishment protocols can be flexibilized infinitely. Also the properties of channels in the literature (and in practice) are very diverse while our model captures a simplified variant related to AEAD. In this section we provide discussions and descriptions for extensions to our model which may help the reader to comprehend our definitional choices. Furthermore the full definition of *Previous Honest Partner* is provided below.

C.1 Strengths and Weaknesses of Matching Conversation-like Partnering

It is important to note that our partnering notion is strong in the sense that it only applies to protocols that protect the integrity of the whole transcript (comparably to the original ACCE model). This results from *matching conversation* based partnering for the channel establishment and the *integrity of ciphertexts* requirement of channel definitions related to AEAD. An implication of this notion is the necessity of strong assumptions on the security of signatures and message authentication codes (for more details we refer the interested reader to [32]). Since the partnering definition is crucial in terms of comparability regarding protocols' security, ambiguous, protocol specific partnering definitions are not an option in our generic framework. Defining partnering via a session key is furthermore not possible since – as argued before – a session key is not a necessary component of fACCE.

In order to slough off technical side effects of the partnering notion on the employed primitives, one can relax the partnering definition by comparing the communicated content (i.e., the messages) instead of the ciphertexts (i.e., the transcript seen on the network), similarly as the RCCA notion [9] relaxes CCA security[24]. Either way, in a model that aims to provide comparable security statements for different protocols, it is important to rely on a generic partnering definition (which our approach ensures).

Furthermore one can consider channel notions for unreliable underlying networks (such that reordering and omissions are tolerated). Such notions could be instantiated with partnering definitions that base on the comparison of the communicated payload instead of matching-conversation style.

C.2 Forward Secrecy within Sessions

While we capture authentication and key compromise impersonation generically, our definitions of forward-secrecy and replay attack resistance are not as generic as they could be: While Noise only reaches forward-secrecy with respect to the long-term keys, it does not implement measures for achieving forward-secrecy within a session with respect to the session state. Having a static state after the handshake also affects the definition of replay attack resistance: if the state within a session would be computed forward securely, the revelation of a state – while replay attacks are possible – would only affect the security of sessions that have the same transcript or of which this transcript is a prefix. In Noise – due to the lack of forward-secrecy in sessions – the revelation of a state affects all sessions with the same *session establishing ciphertext.*

Our model disregards the continuous updates of the channel security. Such updates would make sense when considering forward-secrecy or post compromise security within sessions. Channel notions considering comparable properties have been proposed before and can be used to extend our model in this direction:

[24] This idea stems from discussions with Bertram Poettering at SKECH Workshop 2018.

In order analyze protocols that allow the deterministic update of an established channel key (as in TLS 1.3), Günther and Mazaheri [21] proposed a model that focuses on confidentiality and integrity properties of channels with this feature. We see this as a further flexibilization of ACCE, not necessary for the investigation of Noise because Noise updates channel keys only probabilistically and then keeps the keys static until the end of the session.

Jaeger and Stepanovs [24] and Poettering and Rösler [39] propose strong models for defining security in the presence of adversaries who can reveal the local state. Their model require forward-secrecy and recovery before and after such state reveals respectively. In contrast, our notion considers a session to be trivially broken as soon as the state was revealed. Interestingly, an extension with respect to these properties is not straight forward since these works can disregard long-term keys of users (which need to considered in channel establishment security notions).

C.3 Example Protocol for Model Justification

In Figure 6 we provide an artificial protocol to explain the problem of analyzing security based on multiple isolated property definitions. The depicted protocol deterministically encrypts payload with the first two ciphertexts and only afterwards authenticates the transcript with a signature. Under the assumption that the signature scheme reveals the singing key if an adversary obtains a signature and the respectively utilized random coins, authentication is only reached, if an adversary obtains these random coins after party **B** received ciphertext c_2. However, confidentiality of the payload would be provided even though the randomness was revealed to the adversary even before ciphertext c_0 is sent, as the encryption scheme's security is not harmed by revealing random coins.

Thus, by simply assembling the security counters of an analysis that only considers authentication (without randomness reveal) and an analysis that only considers confidentiality under randomness reveal (without authentication), one would obtain a false security statement for an analysis that considers authentication, confidentiality, and resistance against reveal of randomness.

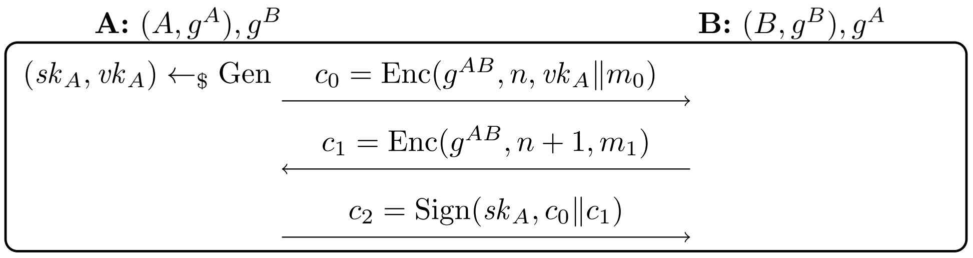

A: $(A, g^A), g^B$
B: $(B, g^B), g^A$
$(sk_A, vk_A) \leftarrow_\$ \text{Gen}$
$c_0 = \text{Enc}(g^{AB}, n, vk_A \| m_0)$
$c_1 = \text{Enc}(g^{AB}, n+1, m_1)$
$c_2 = \text{Sign}(sk_A, c_0 \| c_1)$

Fig. 6: Example protocol that provides client authentication after receiving ciphertext c_2 if randomness is revealed only afterwards (i.e., $\texttt{au}^\texttt{i} = 3, \texttt{eck} > 3$) but that provides confidentiality and no authentication under revealed randomness from the first ciphertext on already (i.e., $\texttt{eck} = 1, \texttt{au}^\texttt{i} = \infty$). The protocol bases on an IND-CCA secure nonce-based symmetric encryption scheme with $\text{Enc}(k, n, m)$ and an SUF-CMA secure signature scheme (that leaks the secret key if the random coins for a signature are revealed).

C.4 Extensions to Authentication Consideration

Authentication and Trust on First Use As it is described in this Appendix, one can extend the granularity of our security definition and of the flexibility of our model arbitrarily. Another option would be a more precise notion of authentication. Our model distinguishes between unauthenticated, authenticated, and KCI resistant protocols, in which the definition leaves it free when each property is reached during a protocol run. Even though unauthenticated parties in a protocol can always be impersonated, often this impersonation needs to be initiated early during the protocol run. One extension to our model would be considering, at which point during a protocol run an active attacker cannot perform an impersonation anymore even though the respective party never authenticates. This property is related to *trust on first* *use* and is especially valuable in settings in which protocol session take a long time. For our setting, we regard the precision of our authentication consideration sufficient.[25]

Reveal of Random Coins during Authentication In our full model we require that for an authenticated ciphertext, the random coins of the receiver of this ciphertext must not be revealed or an active attack against this ciphertext is regarded as a trivial attack. One may think of this as a KCI attack based on the ephemeral secrets of the receiver (instead of its long-term secrets). This trivial attack is tailored to implicit authentication methods related to public key encryption. In contrast, for signature-based authentication (as in Figure 6) it would be necessary that the randomness of the authenticated party's sessions are not revealed in order to rule out trivial attacks. Furthermore, if KCI resistance is not yet reached (i.e., the receiver's long-term key and the sender's long-term key must not be corrupted to avoid trivial attacks) but authentication bases itself on a NIKE among the communicating parties long-term keys, it would not be necessary to forbid the reveal of the receiver session's randomness.

One can indeed treat this issue more flexibly by introducing further variables that control the model. We, however chose to consider only authentication methods related to public key encryption here since it keeps the model comprehensible and in our analysis we consider Noise that bases on this method for authentication.

By always including a NIKE among the long-term secrets of the communicating parties, Noise would be secure even if the described restriction would not be included in the model.

C.5 Set Randomness instead of Reveal Randomness

While our model allows the adversary to reveal random coins, we see alternatives to this definitional choice driven – among other reasons – by valid critique, which we describe below.

Revealing randomness models either the actual leakage of random coins to an adversary or a construction that generates random coins via a random oracle on low entropy input. We believe that both scenarios are motivated by real world threats (e.g., [10, 19]). However – as it might be misunderstood – insecure randomness generators are not depicted by revealing the used random coins to an adversary. The reason for this is: by sampling coins uniformly at random and then providing them to an adversary, the distribution over which these sampling was conducted is still *good*. A weak randomness generator, in contrast, provides 'random' coins from a bad (i.e., non-uniform) distribution. In some real world cases, this distribution might even be chosen by the adversary.

To overcome this gap in the model, we propose to let the adversary set the random coins for chosen sessions itself. The impact of this modification on the model, compared to the eCK model [31], is rather small: mainly cells in Table 3, which define freshness with respect to the corruption of parties and the manipulated (or revealed) randomness, need to be updated. The cells to be updated depend on the `eck` counter that indicates from which point on the session secret is secure even if both session participants' random coins were attacked. Providing full security only with the long-term secrets can indeed be achieved even if, instead of revealing the randomness, the adversary can freely chose and set the random coins of sessions. It can however only be achieved once: if two sessions among two parties A and B both depend only on the security provided by the long-term secrets of A and B, then the adversary can let both sessions run with the exact same secret. In this case, the adversary can trivially attack these sessions. As a result, if for more than one session between two parties, both sides' random coins were set by the adversary, then none of these sessions is declared fresh in any stage.

Full Consideration of Corruptions and Randomness Reveals The consideration of corruptions and reveals of sessions' random coins in our model are limited to these two categories:

1. confidentiality unless one of the two session participant was corrupted (via counter `fs`),
2. confidentiality unless one of the two session participants' random coins were revealed (via counter `eck`), and
3. confidentiality unless the random coins of a session with role ρ were revealed or the peer was corrupted (via counter $\texttt{rl}^{\rho}$).

In general there exist many more combinations with which the model can be extended (e.g., if a stage's secret can only be computed by one session participant asymmetrically such that the secrets of the other participant will not help the adversary) but we see the considered ones as most practically relevant.

[25] Related models such as the MSKE model define authentication less flexible and precise as, e.g., unilateral authentication is fixed to authentication of the initiator.

D Correctness and Partnering Definitions in Pseudocode

For full precision and more clarity, we provide pseudocode descriptions of definitions 2 (correctness), 3 (honest partner), and 11 (previous honest partner) in Figure 7. An fACCE scheme is correct if $\Pr[\text{Correct}(b_{\mathtt{i}}, b_{\mathtt{r}}) \to 1] = 0$ for some $(b_{\mathtt{i}}, b_{\mathtt{r}}) \in \{0,1\}^2$ (depending on for whom the scheme provides authentication).

The correctness game allows an unbounded adversary to play with the scheme's algorithms. An adversary, letting the decryption output an inconsistent pair of message and stage counter (m, ς) (or a decreasing stage counter) in an 'honest execution', wins the correctness game.

The partnering functions compare all sessions' partner identifier, role, associated data, and encrypted and decrypted transcripts with the targeted session's respective values. The corresponding sessions are output as *(previous) hones partners.* As described in definitions 3 and 11, the decryption transcripts must be (or must contain a common) prefix of the encryption transcripts with the (previous) honest partner such that at least one prefix is not empty. After receiving once, there must at most exist one unique honest partner.

```
Game Correct($b_{\mathtt{i}}, b_{\mathtt{r}}$)
$gen \leftarrow 0$; $init \leftarrow 0$
$sk_{\mathtt{i}} \leftarrow \perp$; $pk_{\mathtt{i}} \leftarrow \perp$; $sk_{\mathtt{r}} \leftarrow \perp$; $pk_{\mathtt{r}} \leftarrow \perp$
$st_{\mathtt{i}} \leftarrow \perp$; $st_{\mathtt{r}} \leftarrow \perp$; $\varsigma_{\mathtt{i}}^{\max} \leftarrow 0$; $\varsigma_{\mathtt{r}}^{\max} \leftarrow 0$
$MS_{\mathtt{i}} \leftarrow \perp$; $MS_{\mathtt{r}} \leftarrow \perp$; $T_{\mathtt{i}} \leftarrow \perp$; $T_{\mathtt{r}} \leftarrow \perp$
If $b_{\mathtt{i}} = 1$: $(sk_{\mathtt{i}}, pk_{\mathtt{i}}) \leftarrow_\$ \text{KGen}$
If $b_{\mathtt{r}} = 1$: $(sk_{\mathtt{r}}, pk_{\mathtt{r}}) \leftarrow_\$ \text{KGen}$
Invoke $\mathcal{A}(sk_{\mathtt{i}}, pk_{\mathtt{i}}, sk_{\mathtt{r}}, pk_{\mathtt{r}})$
Abort with 0

Oracle Init($ad$)
If $gen = 0 \lor init = 1 \lor ad \notin \{0,1\}^*$: Abort with 0
$init \leftarrow 1$
$st_{\mathtt{i}} \leftarrow_\$ \text{Init}(sk_{\mathtt{i}}, pk_{\mathtt{r}}, \mathtt{i}, ad)$
$st_{\mathtt{r}} \leftarrow_\$ \text{Init}(sk_{\mathtt{r}}, pk_{\mathtt{i}}, \mathtt{r}, ad)$
Return $st_{\mathtt{i}}, st_{\mathtt{r}}$

Oracle Enc($\rho, m$)
If $init = 0 \lor \rho \notin \{\mathtt{i}, \mathtt{r}\} \lor m \notin \{0,1\}^*$: Abort with 0
$(st_\rho, c, \varsigma) \leftarrow_\$ \text{Enc}(sk_\rho, st_\rho, m)$
If $\varsigma < \varsigma_\rho^{\max}$: Abort with 0
$\varsigma_\rho^{\max} \leftarrow \varsigma$
$MS_\rho \leftarrow MS_\rho \| (m, \varsigma)$
$T_\rho \leftarrow T_\rho \| c$
Return $(st_\rho, c, \varsigma)$

Oracle Dec($\rho$)
If $init = 0 \lor \rho \notin \{\mathtt{i}, \mathtt{r}\}$: Abort with 0
$c \| T_\rho \leftarrow T_\rho$
If $c \notin \{0,1\}^* \setminus \{\epsilon\}$: Abort with 0
$(st_\rho, m, \varsigma) \leftarrow_\$ \text{Dec}(sk_\rho, st_\rho, c)$
$(m^*, \varsigma^*) \| MS_\rho \leftarrow MS_\rho$
If $m \neq \perp \land (m, \varsigma) \neq (m^*, \varsigma^*)$: Abort with 1
```

```
Partner($i, s$)
$partners \leftarrow \emptyset$
For all $j \in [n_P], t \in [n_S]$:
    $a \leftarrow |\pi_j^t.T_d|$
    $b \leftarrow |\pi_i^s.T_d|$
    If $\pi_i^s.pid = j \land \pi_j^t.pid = i$
      $\land \pi_i^s.\rho \neq \pi_j^t.\rho$
      $\land \pi_i^s.ad = \pi_j^t.ad$
      $\land \forall 0 \leq \alpha < a : \pi_i^s.T_e[\alpha] = \pi_j^t.T_d[\alpha]$
      $\land \forall 0 \leq \beta < b : \pi_j^t.T_e[\beta] = \pi_i^s.T_d[\beta]$
      $\land (\pi_i^s.\rho = \mathtt{i} \land a > 0 \lor \pi_i^s.\rho = \mathtt{r} \land b > 0)$:
          $partners \leftarrow partners \cup \{(j, t)\}$
If $\pi_i^s.T_d \neq \perp \land |partners| > 1$:
    Return $\emptyset$
Return $partners$
```

```
PrevPartner($i, s$)
$partners \leftarrow \emptyset$
For all $j \in [n_P], t \in [n_S]$:
    $a \leftarrow |\pi_j^t.T_d|$
    $b \leftarrow |\pi_i^s.T_d|$
    If $\pi_i^s.pid = j \land \pi_j^t.pid = i$
      $\land \pi_i^s.\rho \neq \pi_j^t.\rho$
      $\land \pi_i^s.ad = \pi_j^t.ad$
      $\land \exists 0 \leq a^* \leq a : \forall 0 \leq \alpha < a^* : \pi_i^s.T_e[\alpha] = \pi_j^t.T_d[\alpha]$
      $\land \exists 0 \leq b^* \leq b : \forall 0 \leq \beta < b^* : \pi_j^t.T_e[\beta] = \pi_i^s.T_d[\beta]$
      $\land (\pi_i^s.\rho = \mathtt{i} \land a^* > 0 \lor \pi_i^s.\rho = \mathtt{r} \land b^* > 0)$:
          $partners \leftarrow partners \cup \{(j, t)\}$
Return $partners$
```

Fig. 7: Pseudocode definitions of correctness, honest partnering, and previous honest partnering (see definitions 2, 3, and 11). It is assumed that the '$\|$' symbol is not an element of the ciphertext space and that procedures Partner() and PrevPartner() have access to the respective variables of the security experiment (except for parties' long-term keys and sessions' local states).

E Full Proofs

In this section we detail the full proofs for Noise patterns `N`, `NK`, `NN`, `NX` and `X`, `XK`, `XN`, `XX`.

E.1 N pattern

Theorem 3. *The Noise protocol* `N` *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (\infty, \infty)$*, forward-secrecy* $\mathtt{fs} = \infty$*, KCI resistance* $\mathtt{kc} = (\infty, \infty)$*, one-way-randomness-security* $\mathtt{rl} = (1, \infty)$*, eCK security* $= \infty$*, and replay resistance* $\mathtt{rp} = \infty$*. That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the PRF-ODH assumption* ms-PRF-ODH *with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the* aead *security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{N},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

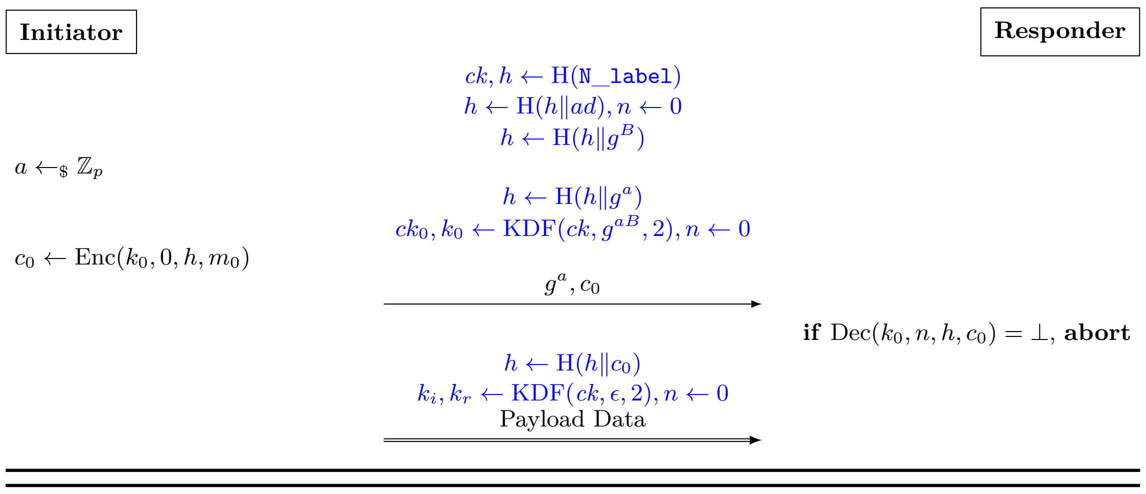

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\mathtt{N\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h \| g^B)$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$ck_0, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$c_0 \leftarrow \mathrm{Enc}(k_0, 0, h, m_0)$
g^a, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 8: Noise Pattern $\mathtt{N} : \leftarrow \mathtt{s}, \ldots, \rightarrow \mathtt{e}, \mathtt{es}$.

Proof.

First, recall that the adversary breaks authentication (and thus, sets $\mathsf{win} \leftarrow 1$) if the received ciphertext was not sent by a session owned by the intended partner, but was successfully decrypted. Since in Noise Pattern `N`, only the initiator sends encrypted ciphertexts, we can restrict our focus to a responder party that successfully decrypts a ciphertext from a non-honest partner. However, by the definition of the security experiment, it is required that if $\mathsf{win} \leftarrow 1$ in the responder session in any given stage, the stage must still be fresh. Since the initiator never authenticates (i.e., $\mathtt{au}^{\mathtt{i}} = \infty$), the adversary injecting the ciphertext from a non-honest partner would cause $\pi_i^s.fr_\varsigma \leftarrow 0 \ \forall \ \varsigma$, and thus $\Pr[\mathsf{win} \leftarrow 1] = 0$. Now we can turn our focus to an adversary attempting to guess the challenge bit b for an initiator session (since the responder session never encrypts anything in Noise Pattern `N`, and thus the behavior of a responder session is independent of its challenge bits). Note that in the Noise Protocol `N` where $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b'), if $\pi_i^s.\rho = \mathtt{i}$, and $\pi_i^s.fr_1 = 1$ then $\mathcal{A}$ cannot issue either a $\mathsf{OCorrupt}(\pi_i^s.pid)$, nor a $\mathsf{ORevealRandomness}(i, s)$ query, as $\mathtt{fs} = \mathtt{eck} = \mathtt{rl}^{\mathtt{r}} = \infty$, and $\mathtt{au}^{\mathtt{r}} = \infty$. We now show that under such restrictions, $\mathcal{A}$ has a negligible advantage in guessing a challenge bit b for the session π_i^s. We begin with the standard fACCE experiment:

$$\mathsf{Adv}^{\mathrm{fACCE}}_{\mathtt{N},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exist two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. The simulator $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Game 3: In this game, we guess the index j of the honest partner π_j^t and abort if $\mathcal{A}$ initializes π_i^s such that $\pi_i^s.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P \big(\mathsf{Adv}(break_3)\big).$$

Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$ in the test session π_i^s and its honest partner (if one exists) by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with an ms-PRF-ODH challenger in the following way:

Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index of the intended partner $\pi_i^s.pid$ of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a ms-PRF-ODH challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{A}$ is not able to issue a $\mathsf{OCorrupt}(j)$ query, as $\mathtt{eck} = \mathtt{fs} = \mathtt{rl}^{\mathtt{r}} = \infty$ and $\mathtt{rl}^{\mathtt{i}} = 1$. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol `N`, the long-term private keys are used to compute the following:

- In sessions where the party acts as the responder: $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

- The other Diffie-Hellman private key x is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of x to complete the computations.
- The other Diffie-Hellman private key x is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary.

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHu oracle provided by the ms-PRF-ODH challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. We note that $\mathtt{fs}, \mathtt{rl}^{\mathtt{r}} = \infty$, and as such $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ never has to answer a $\mathsf{OCorrupt}(j)$ query nor answer a $\mathsf{ORevealRandomness}(i, s)$ query.

Since $\mathtt{rp} = \infty$, however, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ also has to simulate sessions $\pi_j^{t^*}$ where g^v, c_0 (the ciphertext sent by π_i^s) is replayed to $\pi_j^{t^*}$. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ simply sets the computation of ck, k_0 in such sessions to $\widetilde{ck}, \widetilde{k_0}$ to ensure consistency. Thus we have:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ in π_i^s and its honest partner (if one exists). Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$ replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an aead challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when $\varsigma = 1$, (i.e. when using the key $\widetilde{k_0}$) and π_j^t is the honest partner of π_i^s. $\mathcal{B}_{\mathsf{aead}}$ instead queries $\mathrm{Enc}(n, h, m_0, m_1)$ and $\mathrm{Dec}(n, h, c)$ to the aead challenger's oracles.

Since $\widetilde{k_0}$ is a uniformly random and independent value (by **Game 4**), and $\bar{b}$ has an identical distribution to $\pi_i^s.b_1$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

It follows now that any adversary capable of outputting a tuple $(i, s, 1, b')$ such that $b' = \bar{b}$ can be turned into an adversary against the aead security of the AEAD scheme. We also note that the behavior of π_i^s is now entirely independent of the challenge bit b_1, and thus $\mathcal{A}$ can do no better than simply guessing b_1.

Game 7: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_{\mathtt{i}}}$ replaced in **Game 5**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an aead challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when $\varsigma = 2$, (i.e. when using the key $\widetilde{k_{\mathtt{i}}}$) and π_j^t is the honest partner of π_i^s. $\mathcal{B}_{\mathsf{aead}}$ instead queries $\mathrm{Enc}(n, h, m_0, m_1)$ and $\mathrm{Dec}(n, h, c)$ to the aead challenger's oracles.

Since $\widetilde{k_{\mathtt{i}}}$ is a uniformly random and independent value (by **Game 5**), and $\bar{b}'$ has an identical distribution to $\pi_i^s.b_2$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

It follows now that any adversary capable of outputting a tuple $(i, s, 1, b')$ such that $b' = \bar{b}$ can be turned into an adversary against the aead security of the AEAD scheme. We also note that the behavior of π_i^s is now entirely independent of the challenge bit b_2, and thus $\mathcal{A}$ can do no better than simply guessing b_2. Thus:

$$\mathsf{Adv}^{\mathrm{fACCE}}_{\mathtt{N},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

E.2 NK pattern

Theorem 4. *The Noise protocol* NK *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (\infty, 2)$, *forward-secrecy* $\mathtt{fs} = 2$, *KCI resistance* $\mathtt{kc} = (\infty, 2)$, *one-way-randomness-security* $\mathtt{rl} = (1, \infty)$, *eCK security* $\mathtt{eck} = \infty$, *and replay resistance* $\mathtt{rp} = (2, 2)$. *That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the PRF-ODH assumptions* $\mathsf{ms\text{-}PRF\text{-}ODH}$ *and* $\mathsf{nn\text{-}PRF\text{-}ODH}$ *with respect to group* G *and* KDF, $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD, *and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{NK},n_P,n_S,\mathcal{A}} \leq 2 \cdot \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \Big(&\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \\ &\mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}}\Big) + \\ n_P^2 n_S^2 \Big(&2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \\ &\mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}}\Big)\end{aligned}$$

Proof.

We split the analysis into the following three cases:

- **Case A**: $\mathcal{A}$ causes π_i^s, where $\pi_i^s.\rho = \mathtt{i}$, to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 (i.e when π_i^s outputs $\varsigma = 2$ during a decryption call).

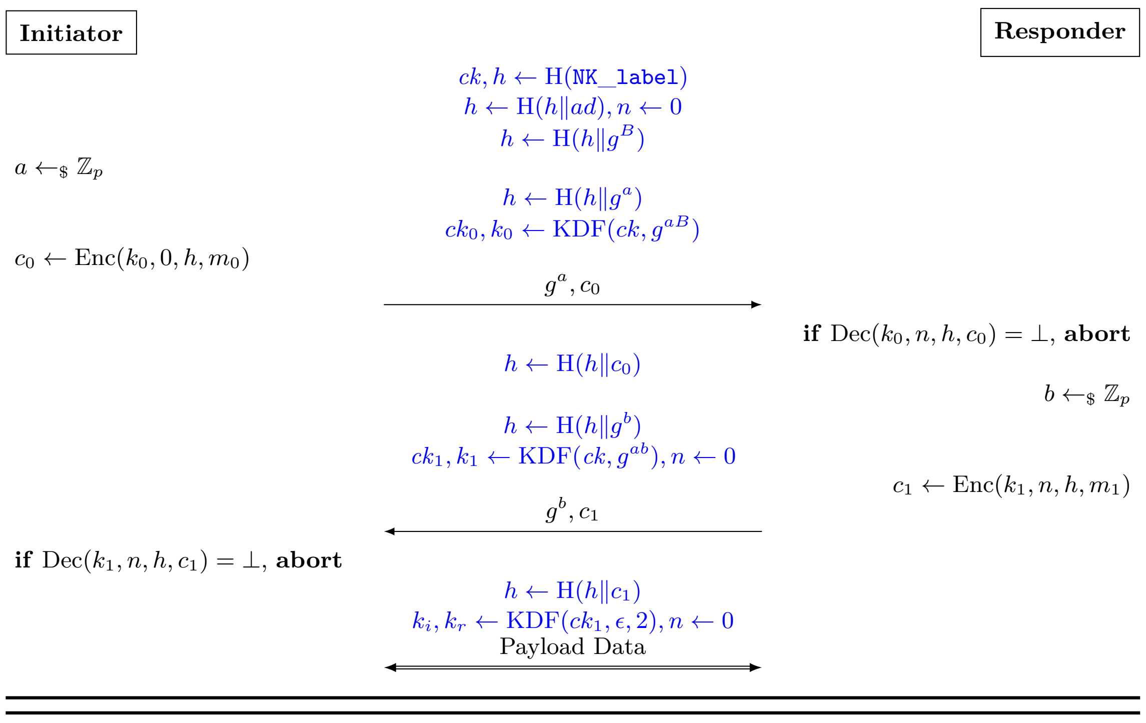

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{NK\_label})$
$h \leftarrow \mathrm{H}(h\|ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h\|g^B)$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h\|g^a)$
$ck_0, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB})$
$c_0 \leftarrow \mathrm{Enc}(k_0, 0, h, m_0)$
g^a, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h\|c_0)$
$b \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h\|g^b)$
$ck_1, k_1 \leftarrow \mathrm{KDF}(ck, g^{ab}), n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_1)$
g^b, c_1
if $\mathrm{Dec}(k_1, n, h, c_1) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h\|c_1)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_1, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 9: Noise Pattern $\texttt{NK} :\leftarrow \texttt{s}, \ldots, \rightarrow \texttt{e}, \texttt{es}$.

- **Case B**: $\mathcal{A}$ causes π_i^s, where $\pi_i^s.\rho = \texttt{r}$, to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the first ciphertext g^a, c_0 (i.e when π_i^s outputs $\varsigma = 1$ during a decryption call).
- **Case C**: $\mathcal{A}$ outputs (i, s, ς', b') and $\mathcal{A}$ does not cause π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 (if $\pi_i^s.\rho = \texttt{i}$) or when π_i^s processes the ciphertext g^a, c_0 (if $\pi_i^s.\rho = \texttt{r}$).

It is clear that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE}}_{\texttt{NK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{fACCE},\mathsf{CA}}_{\texttt{NK},n_P,n_S,\mathcal{A}} &+ \mathsf{Adv}^{\mathsf{fACCE},\mathsf{CB}}_{\texttt{N},n_P,n_S,\mathcal{A}} \\ &+ \mathsf{Adv}^{\mathsf{fACCE},\mathsf{CC}}_{\texttt{NK},n_P,n_S,\mathcal{A}}\end{aligned}$$

We begin by treating **Case A**. In order for π_i^s to set $\mathsf{win} \leftarrow 1$ as defined above, the value of $\pi_i^s.fr_2$ must be 1 and thus $\mathsf{OCorrupt}(\pi_i^s.pid)$ cannot have yet been issued by $\mathcal{A}$ as $\mathsf{au}^{\mathtt{r}} = 2$ and π_i^s will output a stage $\varsigma \leftarrow 2$ when π_i^s decrypts c_1. Similarly, $\mathcal{A}$ cannot have issued a $\mathsf{ORevealRandomness}(i, s)$ query, as one can see when referring to Table 3 as $\mathtt{r}^{\mathtt{i}} = \infty = \texttt{eck}$. Thus we know in this case that $\mathcal{A}$ cannot have issued either a $\mathsf{ORevealRandomness}(i, s)$ nor a $\mathsf{OCorrupt}(\pi_i^s.pid)$ query.

Case A, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\texttt{NK},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case A, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ that triggers if a hash collision occurs. We do so by efficiently defining an algorithm $\mathcal{B}_{\mathsf{coll}}$ that computes all hash values honestly, and aborts if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case A, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \big(\mathsf{Adv}(break_2)\big).$$

Case A, Game 3: In this game, we guess the index j of the intended partner $\pi_i^s.pid$ and abort if $\mathcal{A}$ initializes π_i^s such that $\pi_i^s.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P\big(\mathsf{Adv}(break_3)\big).$$

Case A, Game 4: In this game, we define an abort event $abort_{win}$ if the session π_i^s sets the status $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 (i.e when π_i^s outputs $\varsigma = 2$). Note that the behaviour of a test session π_i^s with $\pi_i^s.\rho = \mathtt{i}$ and $\pi_i^s.b_2 = 1$ does not differ from a session with $\pi_i^s.\rho = \mathtt{i}$ and $\pi_i^s.b_2 = 0$, as the session only uses the challenge bit when it encrypts data, and π_i^s only decrypts data in stage $\varsigma = 1$ (i.e. after sending the first ciphertext g^a, c_0, before it enters stage $\varsigma = 2$ when successfully decrypting g^b, c_1). It follows that the only difference is the advantage $\mathcal{A}$ causes between Game 3 and Game 4 is the advantage $\mathcal{A}$ has in causing $\mathsf{win} = 1$. Note that we do not need to bound the probability that $\mathcal{A}$ guesses $\pi_i^s.b_3$, as we are showing that an adversary that has caused π_i^s to output $\mathsf{win} \leftarrow 1$ when processing the ciphertext g^b, c^1 has negligible chance at reaching the point where the behaviour of the test session π_i^s will differ depending on the test bit b_3. We now bound $\mathsf{Adv}(abort_{win})$. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}(abort_{win}).$$

Case A, Game 5: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger in the following way:

By **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know the index of the intended partner j of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol $\mathtt{NK}$, the long-term private keys are used to compute the following:

– In sessions where the party acts as the responder: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

– The other Diffie-Hellman private key x is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of x to complete the computations.
– The other Diffie-Hellman private key x is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHu oracle provided by the $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. Since in **Game 4**, we abort if $abort_{win}$ is triggered by π_i^s while processing g^b, c_1 (and by **Game 2** $\mathcal{A}$ must output (i, s, ς', b')), by **Game 4** $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ never has to answer a $\mathsf{OCorrupt}(j)$ query. In addition, since $\mathtt{rl}^{\mathtt{i}} = \infty$, $\mathtt{eck} = \infty$, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ also never has to answer a $\mathsf{ORevealRandomness}(i, s)$ query. However $\mathtt{rp}^{\mathtt{i}}, \mathtt{rp}^{\mathtt{r}} = 1$, and $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ also has to simulate sessions $\pi_j^{t^*}$ where g^v, c_0 (the ciphertext sent by π_i^s) is replayed to $\pi_j^{t^*}$. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ sets the computation of ck, k_0 in such sessions to $\widetilde{ck}, \widetilde{k_0}$ to ensure consistency. In addition, within the same session $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ has to potentially simulate the computation of $ck, k_1 \leftarrow \mathrm{KDF}(\widetilde{ck}, g^{b^v})$, where g^b may have been injected by $\mathcal{A}$. In order to compute this, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ queries $\mathsf{ODHv}(\widetilde{ck}, g^b, 2)$ to simulate this computation. Thus we have:

$$\mathsf{Adv}(abort_{win}) \leq \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_5)$$

Case A, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{ab}, 2)$ used to compute $ck, k_1 := \mathrm{KDF}(\widetilde{ck}, g^{ab}, 2)$ in π_i^s and its honest partner (if one exists). Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, and this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case A, Game 7: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$ replaced in **Game 5**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to Enc or Dec queries directed to π_i^s or π_j^t (if an honest partner π_j^t exists) when π_i^s, π_j^t outputs $\varsigma > 1$ (i.e. when using the key $\widetilde{k_0}$ to Encrypt or Decrypt the first ciphertext). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

Since $\widetilde{k_0}$ is a uniformly random and independent value (by **Game 5**), and $\bar{b}$ has an identical distribution to $\pi_i^s.b_1$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_5) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

It follows now that any adversary capable of outputting a tuple $(i, s, 1, b')$ such that $b' = \bar{b}$ can be turned into an adversary against the aead security of the AEAD scheme. We also note that the behavior of π_i^s is now entirely independent of the challenge bit b_1, and thus $\mathcal{A}$ can do no better than simply guessing b_1. Thus:

$$\mathsf{Adv}(break_6) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

Case A, Game 8: In this game, π_i^s will only set $\mathsf{win} \leftarrow 1$ (and thus cause $abort_{win}$ to occur) if $\mathcal{A}$ is able to produce a ciphertext $g^b, c_1 := g^b, \mathsf{AEAD}.\text{Enc}(\widetilde{k_1}, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to both $\text{Enc}(j, t, ad, m_0, m_1)$ and $\text{Dec}(i, s, ad, c)$ queries directed to an honest partner to π_i^s, π_j^t (if such a session exists) and π_i^s when $\pi_i^s.\varsigma = 1$ from $\mathcal{A}$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead queries $\text{Dec}(n, h, c)$ to the AEAD challenger's oracles. Note that if there does not exist a matching session π_j^t to π_i^s that output such a ciphertext, then if c_1 can be successfully decrypted, this will reveal the bit b sampled by the AEAD challenger, allowing $\mathcal{B}_{\mathsf{aead}}$ to break the AEAD security game.

An adversary capable causing $\mathsf{win} \leftarrow 1$ can thus break the aead security of the AEAD scheme. Since $\widetilde{k_1}$ is a uniformly random and independent value (by **Game 6**), this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_7) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

Note that the additional-data field of c_1 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\texttt{NK-\_label}\|ad)\|g^B)\| g^a)\|c_0)\|g^b)$. By **Game 1** we abort the experiment if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 8** there must exist an honest session owned by j that has output c_1. Thus in **Game 8**, $\mathcal{A}$ has no advantage in triggering the event $abort_{win}$ due to π_i^s processing (g^b, c_1), and we find:

$$\begin{aligned}\mathsf{Adv}^{\text{fACCE,CA}}_{\text{NK},n_P,n_S,\mathcal{A}} \le &\mathsf{Adv}^{\mathsf{coll}}_{\text{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\text{KDF},\mathcal{B}\mathsf{prf}} \\ &+ \cdot \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\text{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)\end{aligned}$$

We now treat **Case B**. $\mathcal{A}$ in **Case B** cannot cause π_i^s such that $\pi_i^s.\rho = \texttt{r}$ and $\pi_i^s.fr_1 = 1$ to set $\mathsf{win} \leftarrow 1$, as $\mathsf{au}^{\mathsf{i}} = \infty$. If there does not exist some session π_j^t such that $\pi_i^s.pid = j$ and $\pi_i^s.T_d[1] = \pi_j^t.T_e[1]$, then $\pi_i^s.fr_1 \leftarrow 0$ and thus win is not set to 1.

In the next case, we know that if $\pi_i^s.\rho = \texttt{i}$ ($\pi_i^s.\rho = \texttt{r}$ respectively) then the ciphertexts (g^a, c_0) ((g^b, c_1) respectively) received by π_i^s was the output of an honest partner. We can now treat **Case C**.

Case C, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE,CC}}_{\text{NK},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case C, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exists two evaluations $(in, \text{H}(in))$, $(\hat{in}, \text{H}(\hat{in}))$ such that $in \ne \hat{in}$ but $\text{H}(in) = \text{H}(\hat{in})$. The simulator $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \le \mathsf{Adv}^{\mathsf{coll}}_{\text{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_3)$$

Case C, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(\mathit{break}_1) = n_P n_S \cdot \big(\mathsf{Adv}(\mathit{break}_2)\big).$$

Case C, Game 3: In this game, we guess the index (t, j) of the honest partner π_j^t and abort if π_j^t is not the honest partner of π_i^s. Note that by **Case A** and **Case B**, there *must* exist such an honest partner. Thus:

$$\mathsf{Adv}(\mathit{break}_2) = n_P n_S \cdot \big(\mathsf{Adv}(\mathit{break}_3)\big).$$

At this point, we need to split the analysis into the two following sub-cases. Note that in what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, up to a change in notation.

– **Case C.1**: $\mathcal{A}$ has potentially issued a $\mathsf{ORevealRandomness}(j, t)$ but not issued either a $\mathsf{OCorrupt}(j)$ or $\mathsf{ORevealRandomness}(i, s)$ query.
– **Case C.2**: $\mathcal{A}$ has potentially issued a $\mathsf{OCorrupt}(j)$ query but has not issued either a $\mathsf{ORevealRandomness}(i, s)$ or a $\mathsf{ORevealRandomness}(j, t)$ query. Note that since $\mathcal{A}$ may issue a $\mathsf{OCorrupt}(j)$ query, then $\pi_i^s.\mathit{fr}_1 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i, s, 1, b')$. If $\mathcal{A}$ did not issue a $\mathsf{OCorrupt}(j)$ query, then the security analysis reverts to **Case C.1**.

It is clear that

$$\mathsf{Adv}^{\mathsf{fACCE,CC}}_{\mathtt{NK},n_P,n_S,\mathcal{A}} \leq \mathsf{max}\big(\mathsf{Adv}^{\mathsf{fACCE,CC.1}}_{\mathtt{NK},n_P,n_S,\mathcal{A}}, \mathsf{Adv}^{\mathsf{fACCE,CC.2}}_{\mathtt{NK},n_P,n_S,\mathcal{A}}\big)$$

Case C.1, Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{\mathit{ck}}, \widetilde{k_0}$ in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger as described in **Case A.1, Game 5**. Thus:

$$\mathsf{Adv}(\mathit{break}_3) \leq \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(\mathit{break}_4)$$

Case C.1, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{\mathit{ck}}, g^{ab}, 2)$ used to compute $\mathit{ck}, k_1 := \mathrm{KDF}(\widetilde{\mathit{ck}}, g^{ab}, 2)$ in π_i^s and its honest partner. Since, by **Game 4**, $\widetilde{\mathit{ck}}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{\mathit{ck}}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(\mathit{break}_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(\mathit{break}_5)$$

Case C.1, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{\mathit{ck}}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{\mathit{ck}}, \epsilon, 2)$. Since, by **Game 5**, $\widetilde{\mathit{ck}}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(\mathit{break}_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(\mathit{break}_6)$$

Case C.1, Game 7: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\mathrm{Enc}(i, s, \mathit{ad}, m_0, m_1)$ and $\mathrm{Dec}(j, t, \mathit{ad}, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$ replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ (or $\mathrm{Dec}(n, h, c)$) queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger.

Since $\widetilde{k_0}$ is a uniformly random and independent value (by **Game 4**), and $\bar{b}$ has an identical distribution to $\pi_i^s.b_1$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(\mathit{break}_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(\mathit{break}_7)$$

We note now that if $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b') at the end of the game (where $\varsigma' = 1$), then the behavior of π_i^s is independent of the test bit $\pi_i^s.b_1$ and thus the adversary has no strategy better than simply guessing the random bit $\pi_i^s.b_1$. We continue by showing that the adversary similarly has no advantage in guessing $\pi_i^s.b_3$.

Case C.1, Game 8: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_\mathtt{i}}, \widetilde{k_\mathtt{r}}$ replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_\mathtt{i}}, \widetilde{k_\mathtt{r}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since $\widetilde{k_i}$, $\widetilde{k_r}$ are uniformly random and independent values, this change is sound. Thus,

$$\mathsf{Adv}(break_7) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

In **Case C.1, Game 8**, the behavior of π_i^s is independent of all test bits $\pi_i^s.b_1$, $\pi_i^s.b_2$, $\pi_i^s.b_3$ and thus $\mathcal{A}$ has no advantage in guessing b, nor in triggering $\mathsf{win} \leftarrow 1$. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.1}}_{\mathtt{NK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

We now treat **Case C.2**. By the definition of this sub-case, we know that $\mathcal{A}$ has not issued both a $\mathsf{ORevealRandomness}(i, s)$ and a $\mathsf{ORevealRandomness}(j, t)$ query, but has issued a $\mathsf{OCorrupt}(j)$ query. Since $\mathtt{fs} = 2$ and $\mathtt{rl}^\mathtt{r} = \infty$, by Table 3 any adversary that issues a $\mathsf{OCorrupt}(j)$ sets $\pi_i^s.fr_1 \leftarrow 0$ and outputting $(i, s, 1, b')$ will lose $\mathcal{A}$ the game. As a result, in **Case 2** we cannot prove the security of payload data sent in the first ciphertext, and instead focus on showing that an adversary's advantage in guessing either $\pi_i^s.b_2$ and $\pi_i^s.b_3$ is not negligibly greater than simply returning a randomly sampled bit b'.

Case C.2, Game 4: In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger in the following way:

Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know the index (j, t) of the honest partner session π_j^t. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the ephemeral public-key of session π_i^s, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_j^t. Since we know that the ephemeral key g^b received by π_i^s was output by the honest partner π_j^t, we only need to use the private key u in a single case:

– $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{uB}, 2)$

Since the private key B is an honestly generated long-term private key, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of B to complete the computation. For the honest partner π_j^t, by **Case B** the ephemeral key g^a received by π_j^t was output by the honest partner π_j^t, and $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ needs only replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$ output by the $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.2, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_\mathtt{i}, k_\mathtt{r} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_\mathtt{i}, k_\mathtt{r}$ with uniformly random values $\widetilde{k_\mathtt{i}}, \widetilde{k_\mathtt{r}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.2, Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$ replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ (or $\mathrm{Dec}(n, h, c)$) queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger.

Since $\widetilde{k_1}$ is a uniformly random and independent value (by **Game 4**), and $\bar{b}$ has an identical distribution to $\pi_i^s.b_2$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

We note now that if $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b') at the end of the game (where $\varsigma' = 2$), then the behavior of π_i^s is independent of the test bit $\pi_i^s.b_2$ and thus the adversary has no strategy better than simply guessing the random bit $\pi_i^s.b_2$. We continue by showing that the adversary similarly has no advantage in guessing $\pi_i^s.b_3$.

Case C.2, Game 7: In this game, the challenger flips a bit $\bar{b'}$ and uses $\bar{b'}$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$ replaced in **Game 5**).In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

Since $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values (by **Game 5**), and $\bar{b'}$ has an identical distribution to $\pi_i^s.b_3$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.2, Game 7**, the behavior of π_i^s is independent of the (fresh) test bits $\pi_i^s.b_2$ and $\pi_i^s.b_3$ and thus $\mathcal{A}$ has no advantage in guessing b, nor in triggering $\mathsf{win} \leftarrow 1$. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.2}}_{\mathtt{NK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}\Big)$$

It is clear that $\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.2}}_{\mathtt{NK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.1}}_{\mathtt{NK},n_P,n_S,\mathcal{A}}$), thus $\mathsf{Adv}^{\mathsf{fACCE},\mathsf{CC}}_{\mathtt{NK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.1}}_{\mathtt{NK},n_P,n_S,\mathcal{A}}$)

E.3 NN Pattern

Theorem 5. *The Noise protocol* `NN` *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (\infty, \infty)$*, forward-secrecy* $\mathtt{fs} = 1$*, KCI resistance* $\mathtt{kc} = (\infty, \infty)$*, one-way-randomness-security* $\mathtt{rl} = (\infty, \infty)$*, eCK security* ∞*, and replay resistance* $\mathtt{rp} = (2, 0)$*. That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the PRF-ODH assumption* $\mathsf{nn\text{-}PRF\text{-}ODH}$ *with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{NN},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}\Big)$$

Proof. Note that in what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, except for a change in notation. In order for the behavior of π_i^s to depend upon the test bit $\pi_i^s.b$, $\pi_i^s.fr_\varsigma = 1 \forall$ stages $\varsigma \geq 1$ and thus $\mathsf{ORevealRandomness}(i, s)$ cannot have been issued by $\mathcal{A}$ as $\mathtt{rl}^{\mathtt{i}} = \infty$ and $\mathtt{eck} = \infty$. In addition, since $\mathtt{au}^{\mathtt{r}} = \infty$ and $\mathtt{au}^{\mathtt{i}} = \infty$, if $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b') where $\pi_i^s.fr_1 = 1$ then there

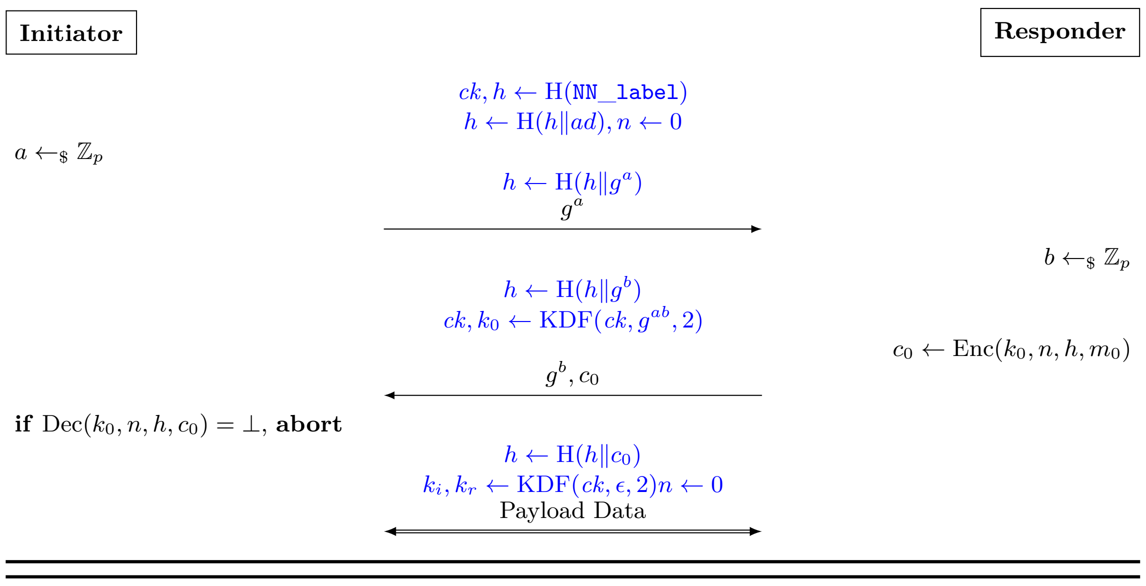

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\mathtt{NN\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
g^a
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, m_0)$
g^b, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2) n \leftarrow 0$
Payload Data

Fig. 10: Noise Pattern $\mathtt{NN} : \rightarrow \mathsf{e}, \leftarrow \mathsf{e}, \mathsf{ee}$.

must exist some session π_j^t such that $\pi_i^s.pid = j$ and $\pi_i^s.T_d[0] = \pi_j^t.T_e[0]$, which means that $\mathcal{A}$ cannot cause π_i^s to set $\mathsf{win} \leftarrow 1$ (i.e. breaking authentication). Finally, the honest partner π_j^t (that must exist) $\mathsf{ORevealRandomness}(j, t)$ cannot have been issued by $\mathcal{A}$ as $\mathtt{rl}^{\mathtt{r}} = \infty$ and $\mathtt{eck} = \infty$.
Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{NN}, n_P, n_S, \mathcal{A}} = \mathsf{Adv}(break_0).$$

Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ that triggers if a hash collision occurs. We do so by defining an algorithm $\mathcal{B}_{\mathsf{coll}}$ that computes all hash values honestly, and aborts if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H}, \mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Game 3: In this game, we guess the index (j, t) of the honest partner π_j^t and abort if π_i^s receives a ciphertext (g^b, c_0) such that $\pi_i^s.T_d[0] \neq \pi_j^t.T_e[0]$. By the argument given above, such a partner must exist, and thus:

$$\mathsf{Adv}(break_2) = n_P n_S \cdot \big(\mathsf{Adv}(break_3)\big).$$

Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger in the following way: $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the ephemeral public-key of session π_i^s, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_j^t. Since we know that the ephemeral key g^b received by π_i^s was output by the honest partner π_j^t, we only need to use the private key u in a single case:

- $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ux}, 2)$

Since we know that the ephemeral key g^b received by π_i^s was output by the honest partner π_j^t $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can simulate this perfectly. In the case that $\mathcal{A}$ forwards g^u or g^v to another session, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ simply uses its own internal knowledge of x to compute $(g^u)^x$. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF}, G, p, \mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ in π_i^s and its honest partner. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$ replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ (or $\mathrm{Dec}(n, h, c)$) queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger.

Since $\widetilde{k_0}$ is a uniformly random and independent value (by **Game 4**), and $\bar{b}$ has an identical distribution to $\pi_i^s.b_1$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

We note now that if $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b') at the end of the game (where $\varsigma' = 1$), then the behavior of π_i^s is independent of the test bit $\pi_i^s.b_1$ and thus the adversary has no strategy better than simply guessing the random bit $\pi_i^s.b_1$. We continue by showing that the adversary similarly has no advantage in guessing $\pi_i^s.b_2$.

Game 7: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$ and $\widetilde{k_{\mathtt{r}}}$ replaced in **Game 5**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ (or $\mathrm{Dec}(n, h, c)$) queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$ or $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger.

Since $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values (by **Game 5**), and $\bar{b}'$ has an identical distribution to $\pi_i^s.b_2$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

We note now that if $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b') at the end of the game (where $\varsigma' = 2$), then the behavior of π_i^s is independent of the test bit $\pi_i^s.b_2$ and thus the adversary has no strategy better than simply guessing the random bit $\pi_i^s.b_2$. In **Game 7**, the behavior of π_i^s is independent of the test bits $\pi_i^s.b_\varsigma\ \forall \varsigma \geq 1$ and thus $\mathcal{A}$ has no advantage in guessing such bits. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{NN},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(&2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\ &+ \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}\Big)\end{aligned}$$

E.4 XN Pattern

Theorem 6. *The Noise protocol* $\mathtt{XN}$ *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (3, \infty)$*, forward-secrecy* $\mathtt{fs} = 2$*, KCI resistance* $\mathtt{kc} = (3, \infty)$*, one-way-randomness-security* $\mathtt{rl} = (\infty, 3)$*, eCK security* $\mathtt{eck} = \infty$*, replay resistance* $\mathtt{rp} = (2, 0)$*. That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in Section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the* sym-ms-PRF-ODH *and* nn-PRF-ODH *assumptions with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{XN},n_P,n_S,\mathcal{A}} \leq 2 \cdot \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} &+ n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big) \\ &+ n_P^2 n_S^2 \cdot \Big(\max\Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big), \\ &\Big(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}} + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)\Big)\end{aligned}$$

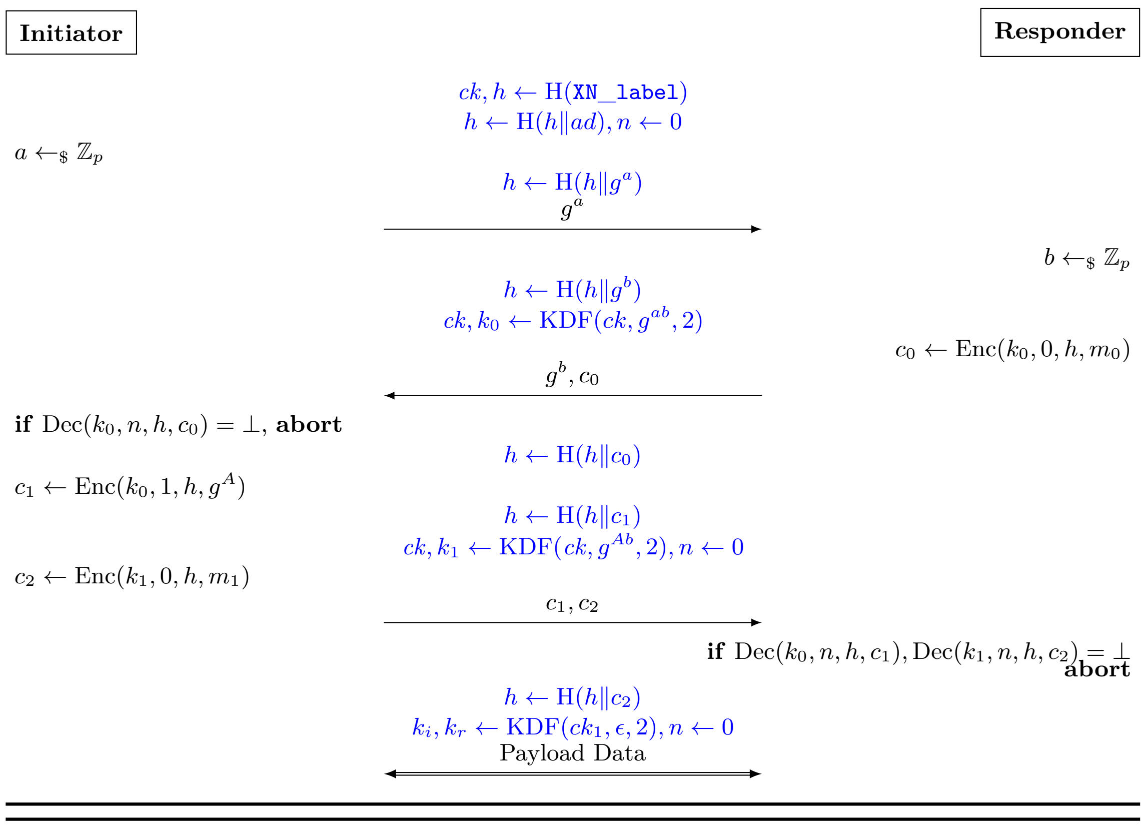

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{XN\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
g^a
$b \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, 0, h, m_0)$
g^b, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$c_1 \leftarrow \mathrm{Enc}(k_0, 1, h, g^A)$
$h \leftarrow \mathrm{H}(h \| c_1)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2), n \leftarrow 0$
$c_2 \leftarrow \mathrm{Enc}(k_1, 0, h, m_1)$
c_1, c_2
if $\mathrm{Dec}(k_0, n, h, c_1), \mathrm{Dec}(k_1, n, h, c_2) = \perp$ **abort**
$h \leftarrow \mathrm{H}(h \| c_2)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_1, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 11: Noise Pattern $\texttt{XN} :\rightarrow \mathsf{e}, \leftarrow \mathsf{e}, \mathsf{ee}, \rightarrow \mathsf{s}, \mathsf{se}$.

Proof.
We split the analysis into the following three cases:

– **Case A**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \texttt{i}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the second ciphertext g^b, c_0 (i.e when π_i^s outputs $\varsigma = 2$ during a decryption call).
– **Case B**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \texttt{r}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the third ciphertext c_1, c_2 (i.e when π_i^s outputs $\varsigma = 3$ during a decryption call).
– **Case C**: $\mathcal{A}$ outputs (i, s, ς', b') and $\mathcal{A}$ does not cause π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_0 (if $\pi_i^s.\rho = \texttt{i}$) or when π_i^s processes the ciphertext c_1, c_2 (if $\pi_i^s.\rho = \texttt{r}$).

We begin by treating **Case A**. $\mathcal{A}$ in **Case A** cannot cause π_i^s such that $\pi_i^s.\rho = \texttt{i}$ and $\pi_i^s.fr_2 = 1$ to set $\mathsf{win} \leftarrow 1$, as $\mathtt{au}^\mathtt{r} = \infty$. There must exist some session π_j^t such that $\pi_i^s.pid = j$ and $\pi_i^s.T_d[1] = \pi_j^t.T_e[1]$, otherwise $\pi_i^s.fr_1 \leftarrow 0$ and thus win is not set to 1.

We now turn to treating **Case B**. In order for π_i^s to set $\mathsf{win} \leftarrow 1$ in **Case B**, $\pi_i^s.fr_3 = 1$ and thus $\mathsf{OCorrupt}(\pi_i^s.pid)$ cannot yet have been issued by $\mathcal{A}$ as $\mathtt{kc}^\mathtt{r} = \infty > \mathtt{au}^\mathtt{i} = 3$, and π_i^s will output $\varsigma = 3$ when decrypting c_1, c_2. Similarly, $\mathcal{A}$ also cannot have issued a $\mathsf{ORevealRandomness}(i, s)$ query as $\mathtt{rl}^\mathtt{r} = 3$ and $\mathtt{eck} = \infty$.

Case B, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE,CB}}_{\texttt{XN}, n_P, n_S, \mathcal{A}} = \mathsf{Adv}(break_0).$$

Case B, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ that triggers if a hash collision occurs. We do so by defining an algorithm $\mathcal{B}_{\mathsf{coll}}$ that computes all hash values honestly, and aborts if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H}, \mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case B, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(\mathit{break}_1) = n_P n_S \cdot \big(\mathsf{Adv}(\mathit{break}_2)\big).$$

Case B, Game 3: In this game, we guess the index (j) of the intended partner π_j^t and abort if $\mathcal{A}$ initializes π_i^s with $\pi_i^s.\mathit{pid} \neq j$. Thus:

$$\mathsf{Adv}(\mathit{break}_2) = n_P \cdot \big(\mathsf{Adv}(\mathit{break}_3)\big).$$

Case B, Game 4: In this game, we define an abort event abort_{win} if the session π_i^s sets the status $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext c_1, c_2 (i.e when π_i^s outputs $\varsigma = 3$). Note that the behavior of a test session π_i^s with $\pi_i^s.\rho = \mathtt{r}$ and $\pi_i^s.b_3 = 1$ does not differ from a session with $\pi_i^s.\rho = \mathtt{r}$ and $\pi_i^s.b_3 = 0$, as the session only uses the challenge bit when it encrypts data, and π_i^s only decrypts data in stage $\varsigma = 3$ (i.e. after sending the ciphertext g^b, c_0, before it enters stage $\varsigma = 3$ when successfully decrypting c_1, c_2). It follows that the only difference is the advantage $\mathcal{A}$ causes between Game 3 and Game 4 is the advantage $\mathcal{A}$ has in causing $\mathsf{win} = 1$, and the advantage that $\mathcal{A}$ has in guessing $\pi_i^s.b_2$. Note that we do not need to bound the probability that $\mathcal{A}$ guesses $\pi_i^s.b_4$, as we are showing that an adversary that has caused π_i^s to output $\mathsf{win} \leftarrow 1$ when processing the ciphertext c_1, c_2 has negligible chance at reaching the point where the behavior of the test session π_i^s will differ depending on the test bit b_4. In what follows, we divide the proof into two further subcases:

- We first bound $\mathsf{Adv}(\mathit{break}_4)$, (the advantage of $\mathcal{A}$ in guessing $\pi_i^s.b_2$, which we refer to as **Case B.1**), and
- We separately bound $\mathsf{Adv}(\mathit{abort}_{win})$ (which we refer to as **Case B.2**).

Thus:

$$\mathsf{Adv}(\mathit{break}_3) \leq \mathsf{Adv}(\mathit{abort}_{win}) + \mathsf{Adv}(\mathit{break}_4).$$

Case B.1, Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{ns\text{-}PRF\text{-}ODH}$ challenger in the following way: $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{ns\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the ephemeral public-key of session π_j^t, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s. In Noise Protocol `XN`, the ephemeral private keys are used to compute the following:

- $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{xv}, 2)$
- $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{yv}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

- The other Diffie-Hellman private key x, y is a value that has been generated by another honest session, or is the long-term private key of another session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of x and y to complete the computations.
- The other Diffie-Hellman private key x is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHv oracle provided by the $\mathsf{ns\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHv}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. Thus:

$$\mathsf{Adv}(\mathit{break}_3) \leq \mathsf{Adv}^{\mathsf{ns\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(\mathit{break}_4)$$

Case B.1, Game 5: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_0}$ replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 4** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ (or $\mathrm{Dec}(n, h, c)$) queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger.

Since $\widetilde{k_0}$ is a uniformly random and independent value (by **Game 4**), and $\bar{b}$ has an identical distribution to $\pi_i^s.b_2$, this change is indistinguishable. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

We note now that if $\mathcal{A}$ terminates and outputs a tuple (i, s, ς', b') at the end of the game (where $\varsigma' = 2$), then the behavior of π_i^s is independent of the test bit $\pi_i^s.b_2$ and thus the adversary has no strategy better than simply guessing the random bit $\pi_i^s.b_2$. Thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}$$

We now turn to bounding the advantage of $\mathcal{A}$ in causing $\mathsf{win} \leftarrow 1$.

Case B.2, Game 5: In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger in the following way:

Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index of the intended partner j of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{A}$ is not able to issue a $\mathsf{OCorrupt}(j)$ query. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol `XN`, the long-term private keys are used to compute the following:

– In sessions where the party acts as the initiator: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

– The other Diffie-Hellman private key x is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of x to complete the computations.
– The other Diffie-Hellman private key x is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHu oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. Similarly, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for the fact that the private key of g^v (the ephemeral public-key of π_i^s) is actually used before the computation of ck, k_1. In particular, it is used earlier in the protocol to compute $ck, k_0 := \mathrm{KDF}(ck, g^{av})$, where g^a may have been contributed by $\mathcal{A}$. In this case, in order to compute ck, k_0, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHv oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHv}(ck, g^a)$, which will output $\mathrm{KDF}(ck, g^{av})$. After processing c_1, c_2, π_i^s will output stage $\varsigma = 3$, and so $\mathcal{A}$ cannot issue a $\mathsf{OCorrupt}(j)$ query *before* π_i^s processes ciphertext c_1, c_2. In addition, since in **Case B**, we abort if $abort_{win}$ is triggered by π_i^s while processing c_1, c_2 (and by **Game 2** $\mathcal{A}$ must output (i, s, ς', b')), by **Game 4** $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ never has to answer a $\mathsf{OCorrupt}(j)$ query. In addition, since $\mathtt{rl}^{\mathtt{r}} = 3$, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ also never has to answer a $\mathsf{ORevealRandomness}(i, s)$ query. Thus we have:

$$\mathsf{Adv}(abort_{win}) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_5)$$

Case B.2, Game 6: In this game, π_i^s will only set $\mathsf{win} \leftarrow 1$ if $\mathcal{A}$ is able to produce a ciphertext c_1, c_2 that decrypts without error. In session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Dec}(i, s, ad, c)$ queries from $\mathcal{A}$ when π_i^s would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead queries $\mathrm{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable causing $\mathsf{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since $\widetilde{k_1}$ is a uniformly random and independent value, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

Note that the additional-data field of c_2 contains $h = \mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathtt{XN\_label}\|ad)\|g^a)\|g^b)\|c_0)\|c_1)$. By **Game 1** we abort the experiment if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 6** we abort if no honest session π_j^t has output c_2. Thus in **Game 6**, $\mathcal{A}$ has no advantage in triggering the event $abort_{win}$ due to π_i^s processing (c_1, c_2). Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB}}_{\mathtt{XN},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{ns\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big)$$

In the next case, we know that the session π_i^s such that $\mathcal{A}$ terminates and outputs (i, s, ς', b') has an honest partner when π_i^s outputs $\varsigma = 1$ (if $\pi_i^s.\rho = \mathtt{i}$ or when π_i^s outputs $\varsigma = 3$ (if $\pi_i^s.\rho = \mathtt{r}$). We can now treat **Case C**.

Case C, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC}}_{\mathtt{XN},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(\mathit{break}_0).$$

Case C, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. The simulator $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(\mathit{break}_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(\mathit{break}_1)$$

Case C, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(\mathit{break}_1) = n_P n_S \cdot \big(\mathsf{Adv}(\mathit{break}_2)\big).$$

Case C, Game 3: In this game, we guess the index (t, j) of the honest partner π_j^t and abort if π_j^t is not the honest partner of π_i^s when π_i^s outputs $\varsigma = 2$ when processing the ciphertext g^b, c_0 (if $\pi_i^s.\rho = \mathtt{i}$) or when π_i^s outputs $\varsigma = 3$ when processing the ciphertext c_1, c_2 (if $\pi_i^s.\rho = \mathtt{r}$). Note that by **Case A** and **Case B**, there *must* exist such an honest partner. Thus:

$$\mathsf{Adv}(\mathit{break}_2) = n_P n_S \cdot \big(\mathsf{Adv}(\mathit{break}_3)\big).$$

At this point, we need to split the analysis into the two following sub-cases. Note that in what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, except for a change in notation.

- **Case C.1**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(i)$ query and an $\mathsf{ORevealRandomness}(j, t)$ query. Note that since $\mathcal{A}$ may issue a $\mathsf{ORevealRandomness}(i, s)$ query, then $\pi_i^s.fr_1 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i, s, 1, b')$. If $\mathcal{A}$ did not issue a $\mathsf{ORevealRandomness}(i, s)$ query, then the security analysis reverts to **Case C.2**.
- **Case C.2**: $\mathcal{A}$ has not issued both a $\mathsf{ORevealRandomness}(i, s)$ and a $\mathsf{ORevealRandomness}(j, t)$ query.

It is clear that

$$\mathsf{Adv}^{\mathsf{fACCE},\mathsf{CC}}_{\mathtt{XN},n_P,n_S,\mathcal{A}} \leq \mathsf{max}\big(\mathsf{Adv}^{\mathrm{fACCE,CC.1}}_{\mathtt{XN},n_P,n_S,\mathcal{A}}, \mathsf{Adv}^{\mathrm{fACCE,CC.2}}_{\mathtt{XN},n_P,n_S,\mathcal{A}}\big)$$

Case C.1, Game 4: In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$ in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger as described in **Case B Game 5**. Thus:

$$\mathsf{Adv}(\mathit{break}_3) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(\mathit{break}_4)$$

Case C.1, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the KDF scheme, and thus:

$$\mathsf{Adv}(\mathit{break}_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(\mathit{break}_5)$$

Case C.1, Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_3$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 4** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since $\widetilde{k_1}$ is a uniformly random and independent value, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

In **Case C.1, Game 6**, the behavior of π_i^s is independent of test bit $\pi_i^s.b_3$. We do the same for $\pi_i^s.b_4$:

Case C.1, Game 7: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_4$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.1, Game 7**, the behavior of π_i^s is independent of the test bits $\pi_i^s.b_\varsigma$, where $\varsigma \geq 3$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits, nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.1}}_{\mathtt{XN},n_P,n_S,\mathcal{A}} \leq\ & \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}} \\ & + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)\end{aligned}$$

We now treat **Case C.2**. **Case C.2, Game 4**: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger in the following way:

Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know the index (j, t) of the honest partner session π_j^t. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the ephemeral public-key of session π_j^t, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s. Since we know that the ephemeral key g^b received by π_i^s was output by the honest partner π_j^t, we only need to use the private key u in a single case:

– $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{uA}, 2)$

Since the private key A is an honestly generated long-term private key, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of A to complete the computation. For the honest partner π_j^t, by **Case B** the ephemeral key g^a received by π_j^t was output by the honest partner π_i^s, and $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ needs only replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$ output by the $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.2, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{aB}, 2)$ used to compute $ck, k_1 := \mathrm{KDF}(\widetilde{ck}, g^{aB}, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.2, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 5**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case C.2, Game 7: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_0}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_0}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.2, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 2$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit.

Case C.2, Game 8: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b}'$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_7) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

In **Case C.2, Game 8**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 3$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit.

Case C.2, Game 9: In this game, the challenger flips a bit $\bar{b}*$ and uses $\bar{b}*$ instead of $\pi_i^s.b_4$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 8** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b}*$ has the same distribution as $\pi_i^s.b_4$, this change is sound. Thus,

$$\mathsf{Adv}(break_8) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_9)$$

In **Case C.2, Game 9**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 2$ and thus $\mathcal{A}$ has no advantage in guessing any challenge bits, nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.2}}_{\mathtt{XN},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}} + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

E.5 NX Pattern

Theorem 7. *The Noise protocol* NX *is flexible-ACCE-secure protocol with authentication levels* $\texttt{au} = (\infty, 2)$*, forward-secrecy* $\texttt{fs} = 2$*, KCI resistance* $\texttt{kc} = (\infty, 2)$*, one-way-randomness-security* $\texttt{rl} = (2, \infty)$*, eCK security* $\texttt{eck} = \infty$ *replay resistance* $\texttt{rp} = (2, 0)$*. That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in Section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the* sym-ms-PRF-ODH *and* nn-PRF-ODH *assumptions with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$
\begin{aligned}
\mathsf{Adv}^{\mathsf{fACCE}}_{\texttt{NX},n_P,n_S,\mathcal{A}} \leq{}& 2 \cdot \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\
&+ \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} \Big) \\
&+ n_P^2 n_S^2 \cdot \Big(\max\Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} \\
&+ \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \Big), \\
&\quad \Big(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} \\
&+ \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \Big) \Big)
\end{aligned}
$$

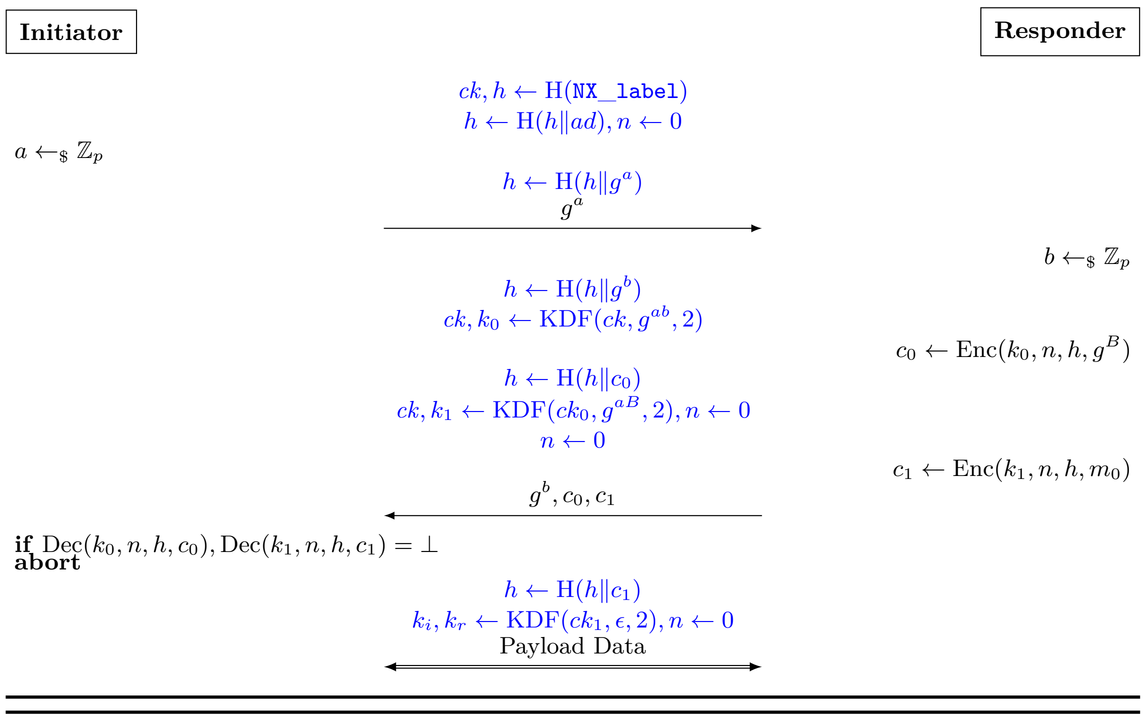

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{NX\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
g^a
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, g^B)$
$h \leftarrow \mathrm{H}(h \| c_0)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck_0, g^{aB}, 2), n \leftarrow 0$
$n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$
g^b, c_0, c_1
if $\mathrm{Dec}(k_0, n, h, c_0), \mathrm{Dec}(k_1, n, h, c_1) = \perp$
abort
$h \leftarrow \mathrm{H}(h \| c_1)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_1, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 12: Noise Pattern NX $:\rightarrow$ e, $\leftarrow$ e, ee, s, es.

Proof.
We split the analysis into the following three cases:

- **Case A**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \texttt{i}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the second ciphertext g^b, c_0, c_1 (i.e when π_i^s outputs $\varsigma = 2$ during a decryption call).
- **Case B**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \texttt{r}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the first ciphertext g^a (i.e when π_i^s outputs $\varsigma = 2$ during a decryption call).

– **Case C**: $\mathcal{A}$ outputs (i, s, ς', b') and $\mathcal{A}$ does not cause π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s outputs $\varsigma = 2$ during a decryption call (if $\pi_i^s.\rho = \mathtt{i}$) or when π_i^s outputs $\varsigma = 1$ during a decryption call (if $\pi_i^s.\rho = \mathtt{r}$).

We begin by treating **Case A**. In order for π_i^s to set $\mathsf{win} \leftarrow 1$ in **Case A**, $\pi_i^s.fr_2 = 1$ and thus $\mathsf{OCorrupt}(\pi_i^s.pid)$ cannot yet have been issued by $\mathcal{A}$ as $\mathtt{kc}^\mathtt{r} = 2 = \mathtt{au}^\mathtt{r} = 2$, and π_i^s will output $\varsigma = 2$ when decrypting g^b, c_0, c_1. Similarly, $\mathcal{A}$ also cannot have issued a $\mathsf{ORevealRandomness}(i, s)$ query as $\mathtt{rl}^\mathtt{r} = \infty$ and $\mathtt{eck} = \infty$.
Case A, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\mathtt{NX},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case A, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ that triggers if a hash collision occurs. We do so by defining an algorithm $\mathcal{B}_{\mathsf{coll}}$ that computes all hash values honestly, and aborts if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case A, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \big(\mathsf{Adv}(break_2)\big).$$

Case A, Game 3: In this game, we guess the party j of the intended partner of the test session π_i^s, and abort if $\pi_i^s.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P \big(\mathsf{Adv}(break_3)\big).$$

Case A, Game 4: In this game, we define an abort event $abort_{win}$ if the session π_i^s sets the status $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_0, c_1 (i.e when π_i^s outputs $\varsigma = 2$). Note that the behavior of a test session π_i^s with $\pi_i^s.\rho = \mathtt{i}$ and $\pi_i^s.b_2 = 1$ does not differ from a session with $\pi_i^s.\rho = \mathtt{i}$ and $\pi_i^s.b_2 = 0$, as the session only uses the challenge bit when it encrypts data, and π_i^s only decrypts data in stage $\varsigma = 2$ (i.e. after sending the ciphertext g^a, before it enters stage $\varsigma = 2$ when successfully decrypting g^b, c_0, c_1). It follows that the only difference is the advantage $\mathcal{A}$ causes between Game 3 and Game 4 is the advantage $\mathcal{A}$ has in causing $\mathsf{win} = 1$, thus $\mathsf{Adv}(break_4) = 0$. Note that we do not need to bound the probability that $\mathcal{A}$ guesses $\pi_i^s.b_2$, as we are showing that an adversary that has caused π_i^s to output $\mathsf{win} \leftarrow 1$ when processing the ciphertext g^b, c_0, c_1 has negligible chance at reaching the point where the behavior of the test session π_i^s will differ depending on the test bit b_2. In what follows, we bound $\mathsf{Adv}(abort_{win})$. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}(abort_{win}).$$

Case A, Game 5: In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger in the following way:

Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index of the intended partner $\pi_i^s.pid$ of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{A}$ is not able to issue a $\mathsf{OCorrupt}(j)$ query. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol $\mathtt{NX}$, the long-term private keys are used to compute the following:

– In sessions where the party acts as the responder: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

– The other Diffie-Hellman private key x is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of x to complete the computations.
– The other Diffie-Hellman private key x is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHu oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. Similarly, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for the fact that the private key of g^v (the ephemeral public-key of π_i^s) is actually used before the computation of ck, k_1. In particular, it is used earlier in the protocol to compute $ck, k_0 := \mathrm{KDF}(ck, g^{av})$, where g^a may have been contributed by $\mathcal{A}$. In this case, in order to compute ck, k_0, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHv oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHv}(ck, g^a)$, which will output $\mathrm{KDF}(ck, g^{av})$. We note that $\mathtt{au}^{\mathtt{r}} = 1$, and while processing g^b, c_0, c_1, π_i^s will output $\varsigma = 2$, and so $\mathcal{A}$ cannot issue a $\mathsf{OCorrupt}(j)$ query before π_i^s processes ciphertext g^b, c_0, c_1. In addition, since in **Case A**, we abort if $abort_{win}$ is triggered by π_i^s while processing g^b, c_0, c_1 (and by **Game 2** $\mathcal{A}$ must output (i, s, ς', b')), by **Game 4** $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ never has to answer a $\mathsf{OCorrupt}(j)$ query. Finally, since $\mathtt{rl}^{\mathtt{i}} = 2$, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ also never has to answer a $\mathsf{ORevealRandomness}(i, s)$ query.Thus we have:

$$\mathsf{Adv}(abort_{win}) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_5)$$

Case A, Game 6: In this game, π_i^s will only set $\mathsf{win} \leftarrow 1$ if $\mathcal{A}$ is able to produce a ciphertext $g^b, c_1, c_2 := g^b, c_1, \mathsf{AEAD}.\mathrm{Enc}(\widetilde{k_1}, n, h, m)$ that decrypts without error. In session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Dec}(i, s, ad, c)$ queries from $\mathcal{A}$ when π_i^s would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead queries $\mathrm{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable causing $\mathsf{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since $\widetilde{k_1}$ is a uniformly random and independent value, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

Note that the additional-data field of c_1 contains $h = \mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathtt{NX\_label}\|ad)\|g^a)\|g^b)\|c_0)$. By **Game 1** we abort the experiment if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 6** we abort if no honest session owned by j has output c_1. Thus in **Game 6**, $\mathcal{A}$ has no advantage in triggering the event $abort_{win}$ due to π_i^s processing (g^b, c_0, c_1). Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\mathtt{NX},n_P,n_S,\mathcal{A}} \leq\ & \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\ & + \mathsf{Adv}^{\mathsf{sym\text{-}mm\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big)\end{aligned}$$

In **Case C**, we know that the ciphertexts g^a, c_0, c_1 received by π_i^s were output by an honest partner. We can now treat **Case B**.

$\mathcal{A}$ in **Case B** cannot cause π_i^s such that $\pi_i^s.\rho = \mathtt{r}$ and $\pi_i^s.fr_1 = 1$ to set $\mathsf{win} \leftarrow 1$, as $\mathtt{au}^{\mathtt{i}} = \infty$. There must exist some session π_j^t such that $\pi_i^s.pid = j$ and $\pi_i^s.T_d[1] = \pi_j^t.T_e[1]$, otherwise $\pi_i^s.fr_1 \leftarrow 0$ and thus win is not set to 1.

We know that in the next case, the ciphertext g^a was the output of some honest partner π_j^t. We can now treat **Case C**.

Case C, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC}}_{\mathtt{NX},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case C, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. The simulator $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_3)$$

Case C, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Case C, Game 3: In this game, we guess the index (t, j) of the honest partner π_j^t and abort if π_j^t is not the honest partner of π_i^s. Note that by **Case A** and **Case B**, there *must* exist such an honest partner. Thus:

$$\mathsf{Adv}(break_2) = n_P n_S \cdot \big(\mathsf{Adv}(break_3)\big).$$

At this point, we need to split the analysis into the two following sub-cases. Note that in what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, except for a change in notation.

– **Case C.1**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(j)$ query and an $\mathsf{ORevealRandomness}(i, s)$ query.
– **Case C.2**: $\mathcal{A}$ has not issued both a $\mathsf{ORevealRandomness}(i, s)$ and a $\mathsf{ORevealRandomness}(j, t)$ query. Note that if $\mathcal{A}$ issues a $\mathsf{OCorrupt}(j)$ query, then $\pi_i^s.fr_1 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i, s, 1, b')$. If $\mathcal{A}$ did not issue a $\mathsf{OCorrupt}(j)$ query, then the security analysis reverts to **Case C.1**.

Case C.1, Game 4: In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$ in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger as described in **Case A, Game 5**. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.1, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the PRF security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.1, Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

In **Case C.1, Game 6**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 2$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits.

Case C.1, Game 7: In this game, the challenger flips a bit $\bar{b'}$ and uses $\bar{b'}$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b'}$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.1, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 2$ and thus $\mathcal{A}$ has no advantage in guessing any challenge bits, nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.1}}_{\mathtt{NX},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

We now treat **Case C.2**.

Case C.2, Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a nn-PRF-ODH challenger in the following way:

Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know the index (j, t) of the honest partner session π_j^t. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a nn-PRF-ODH challenger, embeds the DH challenge keyshare g^u into the ephemeral public-key of session π_i^s, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_j^t. Since we know that the ephemeral key g^b received by π_j^t was output by the honest partner π_i^s, we only need to use the private key u in a single case:

– $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{uB}, 2)$

Since the private key B is an honestly generated long-term private key, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of B to complete the computation. For the honest partner π_j^t, by **Case B** the ephemeral key g^a received by π_j^t was output by the honest partner π_i^s, and $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ needs only replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$ output by the nn-PRF-ODH challenger. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.2, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{aB}, 2)$ used to compute $ck, k_1 := \mathrm{KDF}(\widetilde{ck}, g^{aB}, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.2, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 5**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case C.2, Game 7: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_0}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_1$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.2, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 1$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits.

Case C.2, Game 8: In this game, the challenger flips a bit $\bar{b'}$ and uses $\bar{b'}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output

$\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b}'$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_7) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

In **Case C.2, Game 8**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 2$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits.

Case C.2, Game 9: In this game, the challenger flips a bit $\bar{b}*$ and uses $\bar{b}*$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 8** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b}*$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_8) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_9)$$

In **Case C.2, Game 9**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 1$ and thus $\mathcal{A}$ has no advantage in guessing any challenge bits, nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.2}}_{\mathtt{NX},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(& 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} \\ & + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)\end{aligned}$$

E.6 X pattern

Theorem 8. *The Noise protocol* $\mathtt{X}$ *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (1, \infty)$*, forward-secrecy* $\mathtt{fs} = \infty$*, KCI resistance* $\mathtt{kc} = (\infty, \infty)$*, randomness-security* $\mathtt{rl} = (1, \infty)$*, eCK security* $\mathtt{eck} = 1$*, replay resistance* $\mathtt{rp} = (\infty, \infty)$*. That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the* sym-ms-PRF-ODH *and* nn-PRF-ODH *assumptions with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{X},n_P,n_S,\mathcal{A}} \leq 2 \cdot \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(&\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}mm\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big) \\ + n_P^2 n_S^2 \cdot \Big(\max\Big(&\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{sym\text{-}mm\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big), \\ &\Big(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{G,p,\mathrm{KDF},\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)\Big)\end{aligned}$$

Proof.
We split the analysis into the following two cases:

- **Case A**: $\mathcal{A}$ outputs (i, s, ς', b') such that $\pi_i^s.\rho = \mathtt{r}$ and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the first message flow g^a, c_0, c_1 (i.e., when π_i^s outputs $\varsigma = 1$).
- **Case B**: $\mathcal{A}$ doesn't cause π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the first message flow g^a, c_0, c_1.

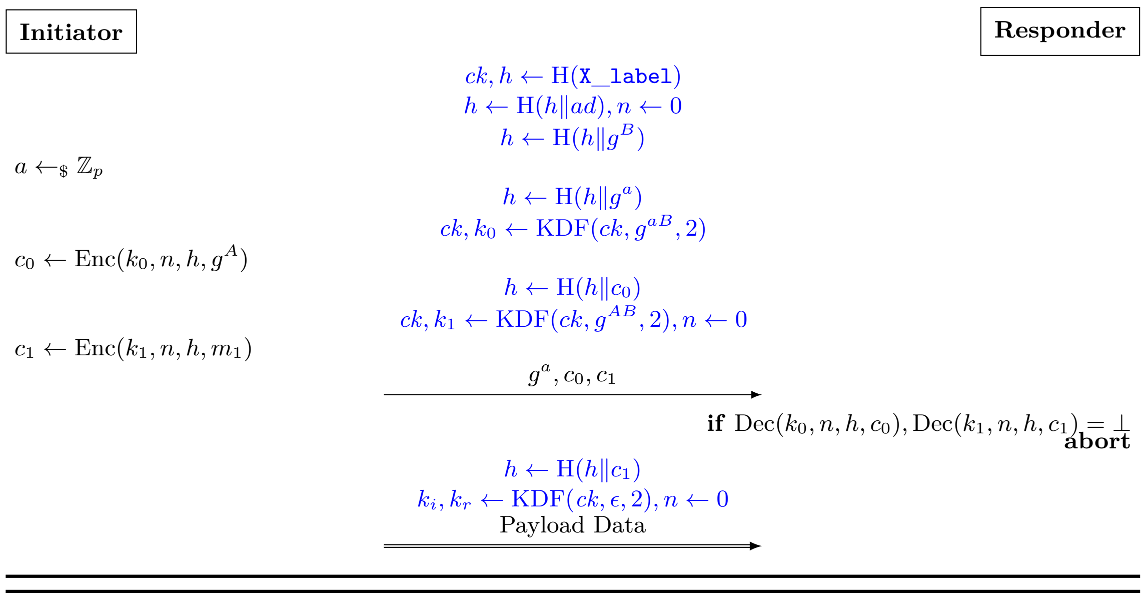

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\mathtt{X\_label})$
$h \leftarrow \mathrm{H}(h\|ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h\|g^B)$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h\|g^a)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, g^A)$
$h \leftarrow \mathrm{H}(h\|c_0)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{AB}, 2), n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_1)$
g^a, c_0, c_1
if $\mathrm{Dec}(k_0, n, h, c_0), \mathrm{Dec}(k_1, n, h, c_1) = \perp$
abort
$h \leftarrow \mathrm{H}(h\|c_1)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 13: Noise Pattern $\mathtt{X} :\leftarrow \mathtt{s}, \ldots, \rightarrow \mathtt{e}, \mathtt{es}, \mathtt{s}, \mathtt{ss}$.

We begin by treating **Case A**. In order for π_i^s to set $\mathsf{win} \leftarrow 1$ in **Case A**, $\pi_i^s.fr_1 = 1$ and thus $\mathsf{OCorrupt}(i)$ cannot have been issued by $\mathcal{A}$ as $\mathtt{kc}^{\mathtt{r}} = \infty > \mathtt{au}^{\mathtt{i}} = 1$, which π_i^s outputs after processing the first message flow, and in addition, $\mathsf{OCorrupt}(\pi_i^s.pid)$ cannot have yet been issued by $\mathcal{A}$ as $\mathtt{au}^{\mathtt{i}} = 1$.
Case A, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE,CA}}_{\mathtt{X},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case A, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ that triggers if a hash collision occurs. We do so by defining an algorithm $\mathcal{B}_{\mathsf{coll}}$ that computes all hash values honestly, and aborts if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case A, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \big(\mathsf{Adv}(break_2)\big).$$

Case A, Game 3: In this game, we guess the party j of the intended partner of the test session π_i^s, and abort if $\pi_i^s.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P \big(\mathsf{Adv}(break_3)\big).$$

Case A, Game 4: In this game, we define an abort event $abort_{win}$ if the session π_i^s sets the status $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^a, c_0, c_1 (i.e when π_i^s outputs $\varsigma = 1$). Note that the behavior of a test session π_i^s with $\pi_i^s.\rho = \mathtt{r}$ and $\pi_i^s.b_1 = 1$ does not differ from a session with $\pi_i^s.\rho = \mathtt{r}$ and $\pi_i^s.b_1 = 0$, as the session only uses the challenge bit when it encrypts data, and π_i^s only decrypts data in stage $\varsigma = 1$ (i.e. outputs $\varsigma = 1$ when successfully decrypting g^a, c_0, c_1). It follows that the only difference is the advantage $\mathcal{A}$ causes between Game 3 and Game 4 is the advantage $\mathcal{A}$ has in causing $\mathsf{win} = 1$, thus $\mathsf{Adv}(break_4) = 0$. Note that we do not need to bound the probability that $\mathcal{A}$ guesses $\pi_i^s.b_2$, as we are showing that an adversary that has caused π_i^s to output $\mathsf{win} \leftarrow 1$ when processing the ciphertext g^a, c_0, c_1 has negligible chance at reaching the point where the behavior of the test session π_i^s will differ depending on the test bit b_2. In what follows, we bound $\mathsf{Adv}(abort_{win})$. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}(abort_{win}).$$

Case A, Game 5: In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$. We do so by defining an algorithm $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a sym-mm-PRF-ODH challenger in the following way:

Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index j of the intended partner j of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a sym-mm-PRF-ODH challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the long-term public-key of party i and give $pk_j = g^u$, $pk_i = g^v$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{A}$ is not able to issue either $\mathsf{OCorrupt}(j)$ or $\mathsf{OCorrupt}(i)$ queries, as $\mathtt{fs} = \infty$, $\mathtt{au} = (1, \infty)$ and $\mathtt{kc} = (\infty, \infty)$. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions t (respectively s) such that party j (respectively party i) must use the private key for computations. In the Noise Protocol $\mathtt{X}$, the long-term private keys are used to compute the following:

– In sessions where the party acts as the initiator: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{uB}, 2)$
– In sessions where the party acts as the responder: $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{au}, 2)$, $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{Au}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

– The other Diffie-Hellman private key (be it a or A) is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of a or A to complete the computations.
– The other Diffie-Hellman private key is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHu (respectively ODHv) oracle provided by the sym-mm-PRF-ODH challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. Thus we have:

$$\mathsf{Adv}(abort_{win}) \leq \mathsf{Adv}^{\mathsf{sym\text{-}mm\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_5)$$

Case A, Game 6: In this game, π_i^s will only set $\mathsf{win} \leftarrow 1$ if $\mathcal{A}$ is able to produce a ciphertext $g^a, c_0, c_1 := g^a, c_0, \mathsf{AEAD}.\mathrm{Enc}(\widetilde{k_1}, n, h, m)$ that decrypts without error. In this game, we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Dec}(j, t, ad, c)$ queries directed to π_i^s when $\varsigma = 0$ from $\mathcal{A}$ (i.e. when using the key $\widetilde{k_1}$). Instead, $\mathcal{B}_{\mathsf{aead}}$ simply forwards these queries to the AEAD challenger.

An adversary capable of causing $\mathsf{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Note that the additional-data field of c_1 contains
$h = \mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathtt{X\_label} \| ad) \| g^B) \| g^a) \| c_0)$. By **Game 1** we abort the experiment if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 6** we abort if no honest session owned by j has output c_1. Thus in **Game 6**, $\mathcal{A}$ has no advantage in triggering the event $abort_{win}$ due to π_i^s processing (g^a, c_0, c_1). Thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}$$

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\mathtt{X},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}mm\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} \Big)$$

In the next case, we know that the ciphertext g^a, c_0, c_1 received by π_i^s were output by an honest partner. We can now treat **Case B**. Note that by **Case A**, if $\pi_i^s.\rho = \mathtt{r}$ we know that there *must* exist some honest partner π_j^t such that $\pi_j^t.T_e[1] = \pi_i^s.T_d[1]$. Note that in **Case B** for π_i^s such that $\pi_i^s.\rho = \mathtt{i}$ and $\pi_i^s.fr_1 = 1$ there must exist some session π_j^t such that $\pi_i^s.pid = j$ and $\pi_i^s.T_e[1] = \pi_j^t.T_d[1]$, otherwise $\pi_i^s.fr_1 \leftarrow 0$.

Case B, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB}}_{\mathtt{X},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case B, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ that triggers if a hash collision occurs. We do so by defining an algorithm $\mathcal{B}_{\mathsf{coll}}$ that computes all hash values honestly, and aborts if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case B, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \big(\mathsf{Adv}(break_1)\big).$$

Case B, Game 3: In this game, we guess the index (t, j) of the honest partner π_j^t and abort if π_j^t is not the honest partner of π_i^s. Note that there *must* exist such an honest partner. Thus:

$$\mathsf{Adv}(break_2) = n_P n_S \cdot \big(\mathsf{Adv}(break_3)\big).$$

At this point, we need to split the analysis into the two following sub-cases. In what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, except for a change in notation.

- **Case B.1**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(i)$ and a $\mathsf{OCorrupt}(j)$ query. Note that if $\mathcal{A}$ issues a $\mathsf{ORevealRandomness}(j, t)$ query at the beginning of the game, then $\pi_i^s.fr_1 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i, s, 1, b')$. If $\mathcal{A}$ did not issue a $\mathsf{ORevealRandomness}(j, t)$ query, then the security analysis reverts to **Case B.2**.
- **Case B.2**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(i)$ query and an $\mathsf{ORevealRandomness}(j, t)$ query.

Case B.1, Game 4: In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$. We do so defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a sym-mm-PRF-ODH challenger in the same way as **Case A, Game 5**. Thus we have:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{sym\text{-}mm\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case B.1, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, ikm, 2)$. Since, by Game 5, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the PRF security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case B.1, Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_1$, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

In **Case B.1, Game 6**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 1$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits.

Case B.1, Game 7: In this game, the challenger flips a bit $\bar{b'}$ and uses $\bar{b'}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in

Game 6 except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\widetilde{b'}$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case B.1, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \ge 1$ and thus $\mathcal{A}$ has no advantage in guessing any challenge bits, nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB.1}}_{\mathtt{X},n_P,n_S,\mathcal{A}} \le \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{sym\text{-}mm\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

We can now treat **Case B.2**.

Case B.2, Game 5: In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$. We do so by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger in the following way:

By the definition of this sub-case (**Case B.2**), we know that $\mathcal{A}$ has not issued a $\mathsf{ORevealRandomness}(i, s)$ query, nor a $\mathsf{OCorrupt}(j)$ query. Additionally, by the analysis of **Case A**, we know that there exists a session π_j^t that received g^a without modification. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of party i and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. However, we must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol `X`, the long-term private keys are used to compute the following:

– In sessions where the party acts as the initiator: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{uB}, 2)$
– In sessions where the party acts as the responder: $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{au}, 2)$, $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{Au}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

– The other Diffie-Hellman private key (be it a, B or u) is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of v or x to complete the computations.
– The other Diffie-Hellman private key is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, the challenger must instead use the ODHu oracle provided by the $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. Thus we have:

$$\mathsf{Adv}(break_4) \le \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_5)$$

Case B.2, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{AB}, 2)$ used to compute $ck, k_1 := \mathrm{KDF}(\widetilde{ck}, g^{AB}, 2)$. Since, by Game 5, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the PRF security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \le \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case B.2, Game 7: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 6**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the PRF security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_6) \le \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_7)$$

Case B.2, Game 8: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_0}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_1$, this change is sound. Thus,

$$\mathsf{Adv}(break_7) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

Case B.2, Game 9: In this game, the challenger flips a bit $\bar{b'}$ and uses $\bar{b'}$ instead of $\pi_i^s.b_1$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 8** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b'}$ has the same distribution as $\pi_i^s.b_1$, this change is sound. Thus,

$$\mathsf{Adv}(break_8) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_9)$$

Case B.2, Game 10: In this game, the challenger flips a bit $\bar{b*}$ and uses $\bar{b*}$ instead of $\pi_i^s.b_2$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 7**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 9** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b*}$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_9) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_{10})$$

In **Case B.2, Game 10**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 1$ and thus $\mathcal{A}$ has no advantage in guessing any challenge bits, nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB.2}}_{\mathtt{X},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(&2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} \\ &+ \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)\end{aligned}$$

E.7 XX pattern

Theorem 9. *The Noise protocol* `XX` *is flexible-ACCE-secure protocol with authentication levels* $\mathtt{au} = (3, 2)$*, forward-secrecy* $\mathtt{fs} = 2$*, KCI resistance* $\mathtt{kc} = (3, 2)$*, randomness-security* $\mathtt{rl} = (2, 3)$*, eCK security* $\mathtt{eck} = \infty$*, replay resistance* $\mathtt{rp} = (2, 0)$*. That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in Section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the* nn-PRF-ODH *and* sym-ms-PRF-ODH *assumptions with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with: More* *precisely:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} \leq\ & 3\cdot\mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S\cdot\Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathrm{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big)+\\ \mathsf{max}\Big(&\big(n_P^2 n_S\big(\mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{aead}}_{\mathrm{AEAD},\mathcal{B}_{\mathsf{aead}}}\big),\\ &\big(n_P^2 n_S^2\big(\mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{aead}}_{\mathrm{AEAD},\mathcal{B}_{\mathsf{aead}}} + 3\cdot\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}}\big)\big)\Big) +\\ &n_P^2 n_S^2\cdot\Big(\mathsf{max}\Big(\big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + 2\cdot\mathsf{Adv}^{\mathsf{aead}}_{\mathrm{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\big),\\ &\quad\big(2\cdot\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + 3\cdot\mathsf{Adv}^{\mathsf{aead}}_{\mathrm{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\big),\\ &\quad\big(3\cdot\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + 4\cdot\mathsf{Adv}^{\mathsf{aead}}_{\mathrm{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\big)\Big)\Big)\end{aligned}$$

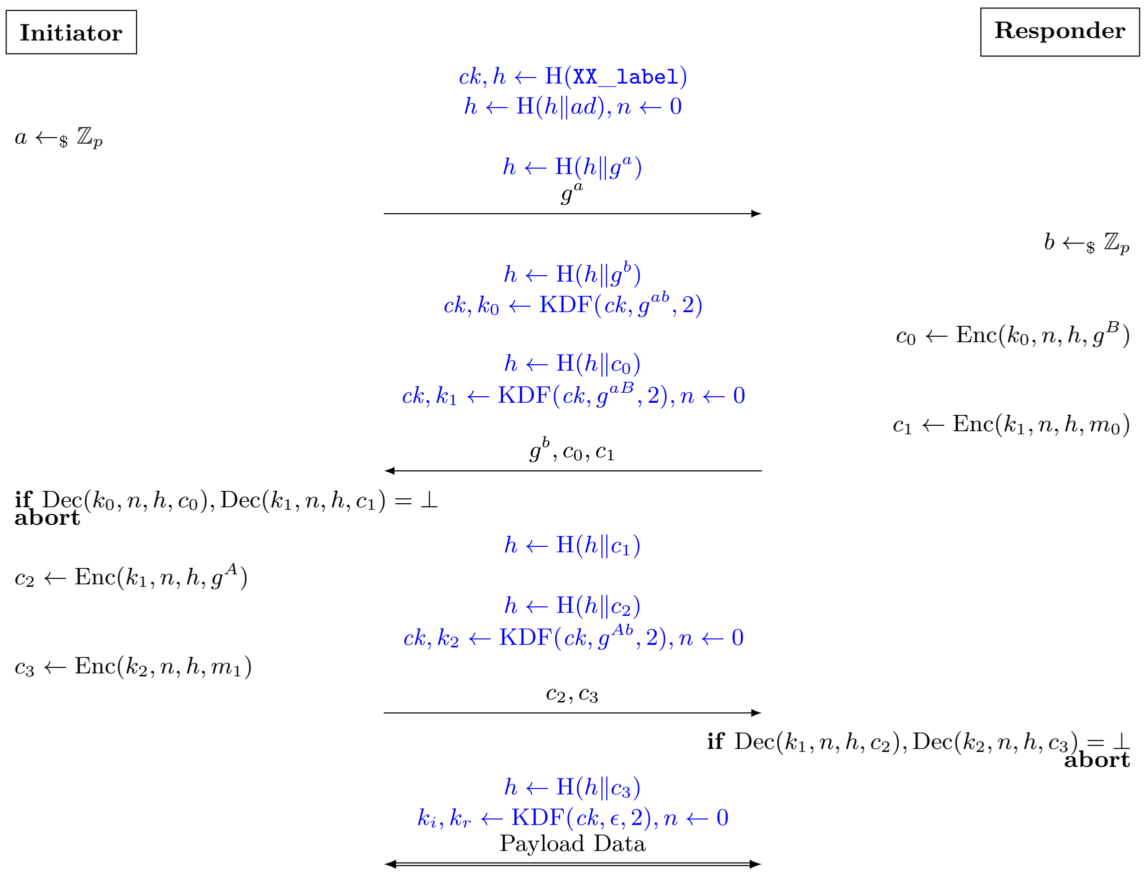

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\mathtt{XX\_label})$
$h \leftarrow \mathrm{H}(h\|ad), n \leftarrow 0$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h\|g^a)$
g^a
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h\|g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, g^B)$
$h \leftarrow \mathrm{H}(h\|c_0)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$
g^b, c_0, c_1
if $\mathrm{Dec}(k_0, n, h, c_0), \mathrm{Dec}(k_1, n, h, c_1) = \perp$
abort
$h \leftarrow \mathrm{H}(h\|c_1)$
$c_2 \leftarrow \mathrm{Enc}(k_1, n, h, g^A)$
$h \leftarrow \mathrm{H}(h\|c_2)$
$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2), n \leftarrow 0$
$c_3 \leftarrow \mathrm{Enc}(k_2, n, h, m_1)$
c_2, c_3
if $\mathrm{Dec}(k_1, n, h, c_2), \mathrm{Dec}(k_2, n, h, c_3) = \perp$
abort
$h \leftarrow \mathrm{H}(h\|c_3)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 14: Noise Pattern $\mathtt{XX} :\rightarrow \mathsf{e}, \leftarrow \mathsf{e}, \mathsf{ee}, \mathsf{s}, \mathsf{es}, \rightarrow \mathsf{s}, \mathsf{se}$.

Proof.
We split the analysis into the following three cases:

- **Case A**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \mathtt{i}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the first ciphertext g^b, c_0, c_1 (i.e., when π_i^s outputs $\varsigma = 1$).
- **Case B**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \mathtt{r}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the second ciphertext c_2, c_3 (i.e., when π_i^s outputs $\varsigma = 2$).
- **Case B**: $\mathcal{A}$ outputs (i, s, ς', b') and $\mathcal{A}$ does not cause π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_0, c_1 (if $\pi_i^s.\rho = \mathtt{i}$) or when π_i^s processes the ciphertext c_2, c_3 (if $\pi_i^s.\rho = \mathtt{r}$).

We begin by treating **Case A**. In order for π_i^s to set $\mathsf{win} \leftarrow 1$ in **Case A**, $\pi_i^s.fr_1 = 1$ and thus $\mathsf{OCorrupt}(\pi_i^s.pid)$ cannot have been issued by $\mathcal{A}$ as $\mathtt{au}^{\mathtt{r}} = 2$ and π_i^s will output $\pi_i^s.\varsigma = 1$ after it processes the ciphertext g^b, c_0, c_1. Also, $\mathcal{A}$ cannot have issued a $\mathsf{ORevealRandomness}(i, s)$ query as $\mathtt{rl}^{\mathtt{i}} = 2$ and $\mathtt{eck} = \infty$.

Case A, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case A, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case A, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Case A, Game 3: In this game, we guess the index (j) of the intended partner $\pi_i^s.pid$ of the test session π_i^s, and abort if $\pi_i^s.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P \cdot \big(\mathsf{Adv}(break_3)\big).$$

Case A, Game 4: In this game, we define an abort event $abort_{win}$ if the session π_i^s sets the status $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_0, c_1. In what follows, we bound the advantage of $\mathcal{A}$ in triggering the event $abort_{win}$.

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}(abort_{win}) + \mathsf{Adv}(break_4).$$

Case A, Game 5: By the definition of this case, and the Noise Protocol $\mathtt{XX}$ we know that $\mathtt{rl}^{\mathtt{i}} = 2$ and $\mathtt{kc}^{\mathtt{i}} = 3 > \mathtt{au}^{\mathtt{r}} = 2$ and π_i^s will only output $\varsigma = 1$ when processing the ciphertext g^b, c_0, c_1. In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$. We do so by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger in the following way:

By the definition of this case (**Case A**), we know that $\mathcal{A}$ has not issued a $\mathsf{ORevealRandomness}(i, s)$ query, nor a $\mathsf{OCorrupt}(j)$ query. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. However, we must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol $\mathtt{XX}$, the long-term private keys are used to compute the following:

– In sessions where the party acts as the initiator: $ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{ub}, 2)$
– In sessions where the party acts as the responder: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{au}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

– The other Diffie-Hellman private key (be it a, or b) is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of a or b to complete the computations.
– The other Diffie-Hellman private key is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, the challenger must instead use the ODHu oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$.

In addition, earlier within the same session π_i^s $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ has to simulate the computation of $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{b^v})$, where g^b may have been injected by $\mathcal{A}$. In order to compute this, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ queries $\mathsf{ODHv}(ck, g^b, 2)$ to simulate this computation. Thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_5)$$

Case A, Game 6: In this game, π_i^s will only set $\mathsf{win} \leftarrow 1$ if $\mathcal{A}$ is able to produce a ciphertext $g^b, c_0, c_1 := g^b, c_0, \mathsf{AEAD}.\mathrm{Enc}(\widetilde{k_1}, n, h, m)$ that decrypts without error. In this game, we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Dec}(j, t, ad, c)$ queries directed to π_i^s when π_i^s would output $\varsigma = 1$ from $\mathcal{A}$ (i.e. when using the key $\widetilde{k_1}$). Instead, $\mathcal{B}_{\mathsf{aead}}$ simply forwards these queries to the AEAD challenger.

An adversary capable of triggering $\mathsf{win} \leftarrow 1$ can break the aead security of the AEAD scheme. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

Note that the additional-data field of c_1 contains $h = \mathrm{H}(\mathrm{H}(\mathrm{H}(\mathtt{XX\_label}\|ad)\|g^a)\|g^b)\|c_0)$. By **Game 1** we abort if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 6** we abort if no honest session owned by j has output c_1. Thus, the probability that $\mathcal{A}$ triggers $abort_{win} = 0$ due to π_i^s processing (g^b, c_0, c_1) is 0. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \left(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\right)$$

Now, we treat **Case B**. In order for π_i^s to set $\mathsf{win} \leftarrow 1$ in **Case B**, $\pi_i^s.fr_3 = 1$ and thus $\mathsf{OCorrupt}(\pi_i^s.pid)$ cannot have yet been issued by $\mathcal{A}$ as $\mathtt{au}^{\mathtt{i}} = 3$ and π_i^s will only output $\varsigma = 3$ after decrypting the ciphertext c_2, c_3. Also, $\mathcal{A}$ cannot have issued a $\mathsf{ORevealRandomness}(i, s)$ query as $\mathtt{rl}^{\mathtt{r}} = 3$ and $\mathtt{eck} = \infty$.

Case B, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case B, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case B, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Case B, Game 3: In this game, we guess the index (j) of the intended partner $\pi_i^s.pid$ of the test session π_i^s, and abort if $\pi_i^s.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P \cdot \big(\mathsf{Adv}(break_3)\big).$$

Case B, Game 4: In this game, we define an abort event $abort_{win}$ if the session π_i^s sets the status $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext c_2, c_3. In what follows, we bound the advantage of $\mathcal{A}$ in triggering the event $abort_{win}$.

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}(abort_{win}) + \mathsf{Adv}(break_4).$$

Case B, Game 5: In this game, we replace the computation of ck, k_2 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_2}$. We do so by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger in the following way:

By the definition of this case (**Case B**), we know that $\mathcal{A}$ has not issued a $\mathsf{ORevealRandomness}(i, s)$ query, nor a $\mathsf{OCorrupt}(j)$ query. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. However, we must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol $\mathtt{XX}$, the long-term private keys are used to compute the following:

– In sessions where the party acts as the initiator: $ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{ub}, 2)$
– In sessions where the party acts as the responder: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{au}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

– The other Diffie-Hellman private key (be it a, or b) is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of a or b to complete the computations.
– The other Diffie-Hellman private key is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, the challenger must instead use the ODHu oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$.

In addition, earlier within the same session π_i^s $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ has to simulate the computation of $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{b^v})$, where g^b may have been injected by $\mathcal{A}$. In order to compute this, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ queries $\mathsf{ODHv}(ck, g^b, 2)$ to simulate this computation. Thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_5)$$

Case B, Game 6: In this game, π_i^s will only set $\mathsf{win} \leftarrow 1$ if $\mathcal{A}$ is able to produce a ciphertext $c_2, c_3 := c_2\ \mathsf{AEAD}.\mathrm{Enc}(\widetilde{k_2}, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s when π_i^s would output $\varsigma = 3$ from $\mathcal{A}$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead queries $\mathrm{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable of triggering $\mathsf{win} \leftarrow 1$ can break the aead security of the AEAD scheme. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

Note that the additional-data field of c_3 contains $h = \mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathtt{XX\_label})\|ad)\|g^a)\|g^b)\|c_0)\|c_1)$. By **Game 1** we abort if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 6** we abort if no honest session owned by j has output c_3. Thus, the probability that $\mathcal{A}$ triggers $abort_{win} = 1$ due to π_i^s processing (c_2, c_3) is 0. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} \leq\ & \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\ & + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big)\end{aligned}$$

Finally, we treat **Case C**.

Case C, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case C, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. The simulator $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_3)$$

Case C, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Case C, Game 3: In this game, we guess the index (t, j) of the honest partner π_j^t and abort if π_j^t is not the honest partner of π_i^s. Note that by **Case A** and **Case B**, there *must* exist such an honest partner. Thus:

$$\mathsf{Adv}(break_2) = n_P n_S \cdot \big(\mathsf{Adv}(break_3)\big).$$

At this point, we need to split the analysis into the three following sub-cases. In what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, except for a change in notation.

– **Case C.1**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(i)$ query and an $\mathsf{ORevealRandomness}(j,t)$ query. Note that if $\mathcal{A}$ issues a $\mathsf{ORevealRandomness}(i,s)$, then $\pi_i^s.fr_1, \pi_i^s.fr_2 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i,s,1,b')$ or $(i,s,2,b')$. If $\mathcal{A}$ did not issue a $\mathsf{ORevealRandomness}(i,s)$ query, then the security analysis reverts to **Case C.3**. Similarly, if $\mathcal{A}$ issues a $\mathsf{OCorrupt}(j)$, then $\pi_i^s.fr2 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i,s,2,b')$. If $\mathcal{A}$ did not issue a $\mathsf{OCorrupt}(j)$ query, then the security analysis reverts to **Case C.2**
– **Case C.2**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(j)$ and a $\mathsf{ORevealRandomness}(i,s)$ query. Note that if $\mathcal{A}$ issues a $\mathsf{ORevealRandomness}(i,s)$, then $\pi_i^s.fr_1, \pi_i^s.fr_2 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i,s,1,b')$ or $(i,s,2,b')$. If $\mathcal{A}$ did not issue a $\mathsf{ORevealRandomness}(i,s)$ query, then the security analysis reverts to **Case C.3**.
– **Case C.3**: $\mathcal{A}$ has not issued both a $\mathsf{ORevealRandomness}(i,s)$ and a $\mathsf{ORevealRandomness}(j,t)$ query.

Case C.1, Game 4: In this game, we replace the computation of ck, k_2 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_2}$ in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger as described in **Case B Game 5**. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.1, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the PRF security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.1, Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i,s,ad,m_0,m_1)$ and $\mathrm{Dec}(j,t,ad,c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Enc}(n,h,m_0,m_1)$ or $\mathrm{Dec}(n,h,c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_2}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

In **Case C.1, Game 6**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 3$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.1, Game 7: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_4$ when responding to $\mathrm{Enc}(i,s,ad,m_0,m_1)$ and $\mathrm{Dec}(j,t,ad,c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n,h,m_0,m_1)$ or $\mathrm{Dec}(n,h,c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b}'$ has the same distribution as $\pi_i^s.b_4$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.1, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 3$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits (and all other challenge bits are not used during the experiment as $\pi_i^s.fr_1 = \pi_i^s.fr_2 = 0$) nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.1}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}$$
$$+\mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

We now treat **Case C.2**.

Case C.2, Game 4: By the definition of this sub-case, we know that $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(j)$ and a $\mathsf{ORevealRandomness}(i,s)$ query. In this game, we replace the computation of ck, k_1 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_1}$ in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ that interacts with a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger as described in **Case A Game 5**. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.2, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_2 with uniformly random values $\widetilde{ck}, \widetilde{k_2}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.2, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 5**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case C.2, Game 7: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.2, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 2$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.2, Game 8: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_2}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_7) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

In **Case C.1, Game 6**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 3$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.2, Game 9: In this game, the challenger flips a bit $\bar{b*}$ and uses $\bar{b*}$ instead of $\pi_i^s.b_4$ when responding to $\mathrm{Enc}(i,s,ad,m_0,m_1)$ and $\mathrm{Dec}(j,t,ad,c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 8** except responding to $\mathrm{Enc}(n,h,m_0,m_1)$ or $\mathrm{Dec}(n,h,c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b'}$ has the same distribution as $\pi_i^s.b_4$, this change is sound. Thus,

$$\mathsf{Adv}(break_8) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_9)$$

In **Case C.2, Game 9**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 2$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits (and the other challenge bit is not used during the experiment as $\pi_i^s.fr_1 = 0$) nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.2}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(&2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} \\ &+\mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)\end{aligned}$$

Finally, we now treat **Case C.3**.

Case C.3, Game 4: By the definition of this sub-case, we know that $\mathcal{A}$ has not issued both a $\mathsf{ORevealRandomness}(i,s)$ and a $\mathsf{ORevealRandomness}(j,t)$ query. In this game, we replace the computation of ck, k_0 with uniformly random and independent values $\widetilde{ck}, \widetilde{k_0}$ in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ interacting with a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger identically to **Case B.2 Game 5**. Thus:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.3, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{aB}, 2)$ used to compute $ck, k_1 := \mathrm{KDF}(\widetilde{ck}, g^{aB}, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.3, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$. Since, by **Game 5**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_2 with uniformly random values $\widetilde{ck}, \widetilde{k_2}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case C.3, Game 7: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 6**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_7)$$

Case C.3, Game 8: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\mathrm{Enc}(i,s,ad,m_0,m_1)$ and $\mathrm{Dec}(j,t,ad,c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to $\mathrm{Enc}(n,h,m_0,m_1)$ or $\mathrm{Dec}(n,h,c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_0}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_1$, this change is sound. Thus,

$$\mathsf{Adv}(break_7) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

In **Case C.3, Game 8**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 1$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.3, Game 9: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_2$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 8** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b}'$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_8) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_9)$$

In **Case C.3, Game 9**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 2$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.3, Game 10: In this game, the challenger flips a bit $\bar{b}''$ and uses $\bar{b}''$ instead of $\pi_i^s.b_3$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 9** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_2}$ is a uniformly random and independent value, and the bit $\bar{b}''$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_9) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_{10})$$

In **Case C.3, Game 10**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 3$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.3, Game 11: In this game, the challenger flips a bit $\bar{b}*$ and uses $\bar{b}*$ instead of $\pi_i^s.b_4$ when responding to $\text{Enc}(i, s, ad, m_0, m_1)$ and $\text{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 7**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 10** except responding to $\text{Enc}(n, h, m_0, m_1)$ or $\text{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b}*$ has the same distribution as $\pi_i^s.b_4$, this change is sound. Thus,

$$\mathsf{Adv}(break_{10}) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_{11})$$

In **Case C.2, Game 10**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 1$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.3}}_{\mathtt{XX},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(3 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}$$
$$+\mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 4 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\Big)$$

E.8 XK pattern

Theorem 10. *The Noise protocol* XK *is flexible-ACCE-secure protocol with authentication levels* $\texttt{au} = (3, 2)$*, forward-secrecy* $\texttt{fs} = 2$*, KCI resistance* $\texttt{kc} = (3, 2)$*, randomness-security* $\texttt{rl} = (1, 3)$*, eCK security* $\texttt{eck} = \infty$*, and replay resistance* $\texttt{rp} = (2, 2)$*. That is, for an adversary* $\mathcal{A}$ *against the flexible ACCE security game (defined in Section 4) one can efficiently define adversaries* $\mathcal{B}_{\mathsf{coll}}$ *against the collision resistance of* H*,* $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ *against the* ms-PRF-ODH*,* nn-PRF-ODH *and* sym-ms-PRF-ODH *assumptions with respect to group* G *and* KDF*,* $\mathcal{B}_{\mathsf{aead}}$ *against the AEAD security of* AEAD*, and* $\mathcal{B}_{\mathsf{prf}}$ *against the PRF security of* KDF *with:*

$$
\begin{aligned}
\mathsf{Adv}^{\mathsf{fACCE}}_{\texttt{XK},n_P,n_S,\mathcal{A}} \leq{}& 3 \cdot \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} \\
&+2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big) \\
&+n_P^2 n_S^2 \cdot \Big(\mathsf{max}\Big(\big(3 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 4 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}}\big), \\
&\quad \big(\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + 2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\big), \\
&\quad \big(2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + 3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\big)\Big)\Big)
\end{aligned}
$$

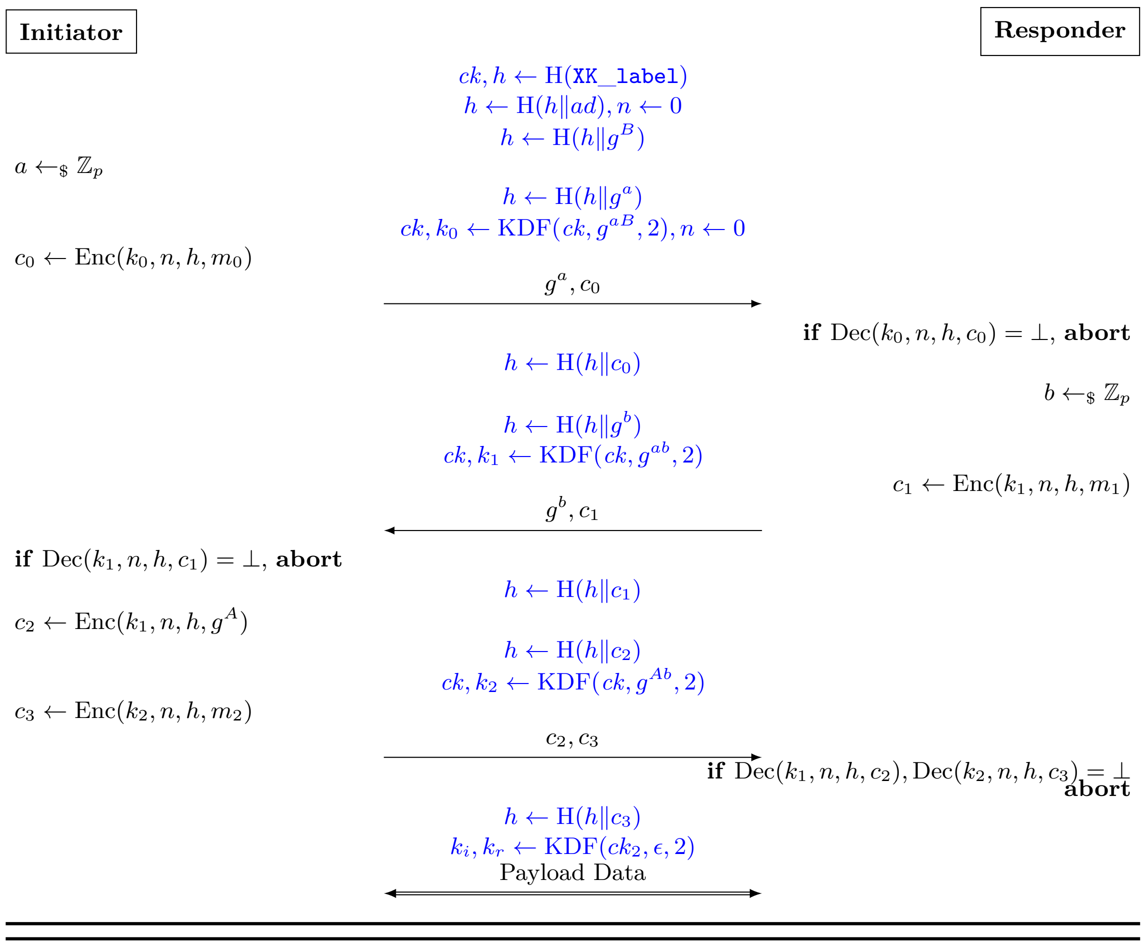

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{XK\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h \| g^B)$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, m_0)$
g^a, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_1)$
g^b, c_1
if $\mathrm{Dec}(k_1, n, h, c_1) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_1)$
$c_2 \leftarrow \mathrm{Enc}(k_1, n, h, g^A)$
$h \leftarrow \mathrm{H}(h \| c_2)$
$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2)$
$c_3 \leftarrow \mathrm{Enc}(k_2, n, h, m_2)$
c_2, c_3
if $\mathrm{Dec}(k_1, n, h, c_2), \mathrm{Dec}(k_2, n, h, c_3) = \perp$ **abort**
$h \leftarrow \mathrm{H}(h \| c_3)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_2, \epsilon, 2)$
Payload Data

Fig. 15: Noise Pattern XK $: \leftarrow \mathsf{s}, \ldots, \rightarrow \mathsf{e}, \mathsf{es} \leftarrow \mathsf{e}, \mathsf{ee} \rightarrow \mathsf{s}, \mathsf{se}$.

Proof.
We split the analysis into the following three cases:

- **Case A**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \mathtt{i}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the second ciphertext g^b, c_1 (i.e., when π_i^s outputs $\varsigma = 2$).
- **Case B**: $\mathcal{A}$ outputs (i, s, ς', b') (where $\pi_i^s.\rho = \mathtt{r}$) and $\mathcal{A}$ causes π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the third ciphertext c_2, c_3 (i.e., when π_i^s outputs $\varsigma = 3$).
- **Case B**: $\mathcal{A}$ outputs (i, s, ς', b') and $\mathcal{A}$ does not cause π_i^s to set $\mathsf{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 (if $\pi_i^s.\rho = \mathtt{i}$) or when π_i^s processes the ciphertext c_2, c_3 (if $\pi_i^s.\rho = \mathtt{r}$).

We begin by treating **Case A**.

Case A, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\mathtt{XK},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case A, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case A, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Case A, Game 3: In this game, we guess the index j of the honest partner π_j^t and abort if $\mathcal{A}$ initializes π_i^s such that $\pi_i^s.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P\big(break_3\big).$$

Case A, Game 4: In this game, we define an abort event $abort_{win}$ if the session π_i^s sets the status $\mathsf{win} \leftarrow 1$ when π_i^s processes the second ciphertext g^b, c_1. In what follows, we bound the advantage of $\mathcal{A}$ in triggering the event $abort_{win}$.

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}(abort_{win}) + \mathsf{Adv}(break_4).$$

Case A, Game 5: Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index of the intended partner $\pi_i^s.pid$ of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party j, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_i^s and give $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{A}$ is not able to issue a $\mathsf{OCorrupt}(j)$ query, as π_i^s will output $\varsigma = 2$ *after* π_i^s processes the ciphertext g^b, c_1, and $\mathsf{au}^{\mathsf{r}} = 2$. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol $\mathtt{XK}$, the long-term private keys are used to compute the following:

- In sessions where the party acts as the initiator: $ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$
- In sessions where the party acts as the responder: $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

- The other Diffie-Hellman private key x is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of x to complete the computations.
- The other Diffie-Hellman private key x is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHu oracle provided by the $\mathsf{ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. We note that $\mathsf{au}^{\mathsf{r}} = 2$, and only after processing (g^b, c_1) will π_i^s output $\varsigma = 2$, and so $\mathcal{A}$ cannot issue a $\mathsf{OCorrupt}(j)$ query before π_i^s processes ciphertext g^b, c_1. In addition, since in **Case A**, π_i^s sets win and we abort if $abort_{win}$ is triggered by π_i^s while processing g^b, c_1 by **Game 4** $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ never has to answer a $\mathsf{OCorrupt}(j)$ query. In addition, since

$\mathtt{rl}^{\mathtt{r}} = 3$ and $\mathtt{eck} = \infty$, in **Case A** $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ also never has to answer a $\mathsf{ORevealRandomness}(i, s)$ query. Thus we have:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_6)$$

Case A, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{ab}, 2)$ used to compute $ck, k_1 := \mathrm{KDF}(\widetilde{ck}, g^{ab}, 2)$. Since, by **Game 5**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case A, Game 7: In this game, π^s_i will only set $\mathsf{win} \leftarrow 1$ if $\mathcal{A}$ is able to produce a ciphertext $g^b, c_1 := g^b, \mathsf{AEAD}.\mathrm{Enc}(\widetilde{k_1}, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Dec}(n, h, c)$ queries directed to π^s_i when π^s_i would output $\varsigma = 2$ from $\mathcal{A}$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead queries $\mathrm{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable of causing $\mathsf{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since $\widetilde{k_1}$ is a uniformly random and independent value, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

Note that the additional-data field of c_1 contains $h = \mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathtt{XK\_label}\|ad)\|g^B)\|g^a)\|c_0)\|g^b)$. By **Game 1** we abort the experiment if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 4** we abort if no honest session owned by j has output c_1. Thus, the probability that $\mathcal{A}$ triggers $abort_{win} = 1$ due to π^s_i processing (g^b, c_1) is 0. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CA}}_{\mathtt{XK},n_P,n_S,\mathcal{A}} \leq \; & \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\ & + \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}\Big)\end{aligned}$$

We now treat **Case B**.

Case B, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB}}_{\mathtt{XK},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case B, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_1)$$

Case B, Game 2: In this game, we guess the index (i, s) of the session π^s_i, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Case B, Game 3: In this game, we guess the index j of the honest partner π^t_j and abort if $\mathcal{A}$ initializes π^s_i such that $\pi^s_i.pid \neq j$. Thus:

$$\mathsf{Adv}(break_2) = n_P\big(break_3\big).$$

Case B, Game 4: In this game, we define an abort event $abort_{win}$ if the session π^s_i sets the status $\mathsf{win} \leftarrow 1$ when π^s_i processes the ciphertext c_2, c_3. In what follows, we bound the advantage of $\mathcal{A}$ in triggering the event $abort_{win}$.

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}(abort_{win}) + \mathsf{Adv}(break_4).$$

Case B, Game 5: In this game, we replace the computation of $ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{Ab})$ with uniformly random and independent values $(\widetilde{ck}, \widetilde{k_2})$ in the test session π^s_i and its honest partner. Note that by **Game 2**, we know at the beginning of the experiment the index of session π^s_i such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index (j,t) of the honest partner π_j^t of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the long-term public-key of party i, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_j^t and give $pk_i = g^u$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ never has to answer a $\mathsf{OCorrupt}(i)$ query nor a $\mathsf{ORevealRandomness}(j,t)$ query. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions s such that party i must use the private key for computations. In the Noise Protocol `XK`, the long-term private keys are used to compute the following:

- In sessions where the party acts as the initiator: $ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$
- In sessions where the party acts as the responder: $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in two ways:

- The other Diffie-Hellman private key x is a value that has been generated by another honest session. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of x to complete the computations.
- The other Diffie-Hellman private key x is a value that is unknown to $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHu oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\mathrm{KDF}(ck, X^u)$. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for the fact that the private key of g^v (the ephemeral public-key of π_i^s) is actually used before the computation of ck, k_2. In particular, it is used earlier in the protocol to compute $ck, k_0 := \mathrm{KDF}(ck, g^{av})$, where g^a may have been contributed by $\mathcal{A}$. In this case, in order to compute ck, k_0, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must instead use the ODHv oracle provided by the $\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}$ challenger, specifically querying $\mathsf{ODHv}(ck, g^a)$, which will output $\mathrm{KDF}(ck, g^{av})$. Thus we have:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_6)$$

Case B, Game 6: In this game, π_i^s will only set $\mathsf{win} \leftarrow 1$ if $\mathcal{A}$ is able to produce a ciphertext $c_2, c_3 := c_2, \mathsf{AEAD}.\mathrm{Enc}(\widetilde{k_2}, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Dec}(n,h,c)$ queries directed to π_i^s when π_i^s would output $\varsigma = 3$ from $\mathcal{A}$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead queries $\mathrm{Dec}(n,h,c)$ to the AEAD challenger's oracles.

An adversary capable of causing $\mathsf{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since $\widetilde{k_1}$ is a uniformly random and independent value, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

Note that the additional-data field of c_3 contains $h = \mathrm{H}(\mathrm{HH}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\mathrm{H}(\texttt{XK\_label} \| ad) \| g^B) \| g^a) \| c_0) \| g^b) \| c_1) \| c2)$. By **Game 1** we abort the experiment if $\mathcal{A}$ causes a hash-collision to occur, and by **Game 4** we abort if no honest session owned by j has output c_3. Thus, the probability that $\mathcal{A}$ triggers $abort_{win} = 1$ due to π_i^s processing (c_2, c_3) is 0. Thus:

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CB}}_{\mathtt{XK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S \cdot \Big(\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}}\Big)$$

In the next case, we know that the ciphertexts $(g^a,\ c_0)$, $(g^b,\ c_1)$, $(c_2,\ c_3)$ received by π_i^s and π_j^t were output by honest partners. We can now treat **Case C**.

Case C, Game 0: This is the standard fACCE experiment.

$$\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC}}_{\mathtt{XK},n_P,n_S,\mathcal{A}} = \mathsf{Adv}(break_0).$$

Case C, Game 1: In this game we define an abort event $abort_{\mathsf{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exists two evaluations $(in, \mathrm{H}(in))$, $(\hat{in}, \mathrm{H}(\hat{in}))$ such that $in \neq \hat{in}$ but $\mathrm{H}(in) = \mathrm{H}(\hat{in})$. The simulator $\mathcal{B}_{\mathsf{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\mathsf{Adv}(break_0) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + \mathsf{Adv}(break_3)$$

Case C, Game 2: In this game, we guess the index (i, s) of the session π_i^s, and abort if $\mathcal{A}$ terminates and outputs $(i^*, s^*, \varsigma', b')$ such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\mathsf{Adv}(break_1) = n_P n_S \cdot \big(\mathsf{Adv}(break_2)\big).$$

Case C, Game 3: In this game, we guess the index (t, j) of the honest partner π_j^t and abort if π_j^t is not the honest partner of π_i^s. Note that by **Case A** and **Case B**, there *must* exist such an honest partner. Thus:

$$\mathsf{Adv}(break_2) = n_P n_S\big(break_3\big).$$

At this point, we need to split the analysis into the three following sub-cases. Note that in what follows, we assume without loss of generality that π_i^s is the initiator session. The analysis where π_i^s is the responder session follows identically, except for a change in notation.

- **Case C.1**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(j)$ query and an $\mathsf{ORevealRandomness}(i, s)$ query. This allows us to prove the security of all stages ciphertexts.
- **Case C.2**: $\mathcal{A}$ has not issued both a $\mathsf{OCorrupt}(i)$ query and an $\mathsf{ORevealRandomness}(j, t)$ query. Note that if $\mathcal{A}$ issues a $\mathsf{ORevealRandomness}(i, s)$ query at the beginning of the game, then $\pi_i^s.fr_1, \pi_i^s.fr_2 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i, s, 1, b')$ or $(i, s, 2, b')$. This allows us to prove the security of ciphertexts belonging to stages $\varsigma \geq 3$. Note that if $\mathcal{A}$ additionally did not issue a $\mathsf{ORevealRandomness}(j, t)$ query, then the security analysis reverts to **Case C.3**, since $\pi_i^s.fr_2 = 1$, and we need to capture the security of the additional stage's ciphertext.
- **Case C.3**: $\mathcal{A}$ has not issued both a $\mathsf{ORevealRandomness}(i, s)$ and a $\mathsf{ORevealRandomness}(j, t)$ query. Note that if $\mathcal{A}$ issues a $\mathsf{OCorrupt}(j)$, then $\pi_i^s.fr_1 \leftarrow 0$, and thus $\mathcal{A}$ has no advantage in outputting $(i, s, 1, b')$. This allows us to prove the security of ciphertexts belonging to stages $\varsigma \geq 3$. Note that if $\mathcal{A}$ additionally did not issue a $\mathsf{OCorrupt}(j)$ query, then the security analysis reverts to **Case C.1** since $\pi_i^s.fr_1 = \pi_i^s.fr_2 = 1$, and we need to capture the security of the additional stages' ciphertext.

Case C.1, Game 4: In this game, we replace the computation of $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB})$ with uniformly random and independent values $(\widetilde{ck}, \widetilde{k_0})$ in the test session π_i^s and its honest partner. This game proceeds identically to **Case A Game 5**. Thus we have:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.1, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{ab}, 2)$ used to compute $ck, k_1 := \mathrm{KDF}(\widetilde{ck}, g^{ab}, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_1 with uniformly random values $\widetilde{ck}, \widetilde{k_1}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.1, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$. Since, by **Game 5**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_2 with uniformly random values $\widetilde{ck}, \widetilde{k_2}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case C.1, Game 7: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 6**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_7)$$

Case C.1, Game 8: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_1$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ (i.e. when using the key $\widetilde{k_0}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_0}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_1$, this change is sound. Thus,

$$\mathsf{Adv}(\mathit{break}_7) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(\mathit{break}_8)$$

In **Case C.1, Game 8**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 1$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.1, Game 9: In this game, the challenger flips a bit $\bar{b'}$ and uses $\bar{b'}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 8** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b'}$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(\mathit{break}_8) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(\mathit{break}_9)$$

In **Case C.1, Game 9**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 2$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.1, Game 10: In this game, the challenger flips a bit $\bar{b''}$ and uses $\bar{b''}$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 9** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_2}$ is a uniformly random and independent value, and the bit $\bar{b''}$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(\mathit{break}_9) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(\mathit{break}_{10})$$

In **Case C.1, Game 10**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 3$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.1, Game 11: In this game, the challenger flips a bit $\bar{b}*$ and uses $\bar{b}*$ instead of $\pi_i^s.b_4$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 7**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 10** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b}*$ has the same distribution as $\pi_i^s.b_4$, this change is sound. Thus,

$$\mathsf{Adv}(\mathit{break}_{10}) \le \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(\mathit{break}_{11})$$

In **Case C.2, Game 10**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \ge 1$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.1}}_{\mathtt{XK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(&4 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\ &+\mathsf{Adv}^{\mathsf{ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 3 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}\Big)\end{aligned}$$

We now treat **Case C.2**.

Case C.2, Game 4: In this game, we replace the computation of $ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{Ab})$ with uniformly random and independent values $(\widetilde{ck}, \widetilde{k_2})$ in the test session π_i^s and its honest partner. This game proceeds identically to **Case B Game 5**. Thus we have:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.2, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.2, Game 6: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 5** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_2}$ is a uniformly random and independent value, and the bit $\bar{b''}$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_6)$$

In **Case C.2, Game 6**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 3$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.2, Game 7: In this game, the challenger flips a bit $\bar{b'}$ and uses $\bar{b'}$ instead of $\pi_i^s.b_4$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b'}$ has the same distribution as $\pi_i^s.b_4$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.2, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 3$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits (and the other challenge bits are not used during the experiment as $\pi_i^s.fr_1 = \pi_i^s.fr_2 = 0$) nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.2}}_{\mathtt{XK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(&2 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\ &+\mathsf{Adv}^{\mathsf{sym\text{-}ms\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}\mathsf{prf}}\Big)\end{aligned}$$

We now treat **Case C.3**.

Case C.3, Game 4: In this game, we replace the computation of $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{ab})$ with uniformly random and independent values $(\widetilde{ck}, \widetilde{k_1})$ in the test session π_i^s and its honest partner. Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ς', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index (j, t) of the honest partner π_j^t of the session π_i^s. Thus, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ initializes a $\mathsf{nn\text{-}PRF\text{-}ODH}$ challenger, embeds the DH challenge keyshare g^u into the ephemeral public-key of session π_i^s, embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_j^t and give $pk_i = g^u$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ never has to answer a $\mathsf{ORevealRandomness}(j, t)$ query nor a $\mathsf{ORevealRandomness}(j, t)$ query. However, $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must account for all sessions such that $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ must use the private key for computations. In the Noise Protocol `XK`, the ephemeral private keys are used to compute the following:

– In sessions where the party acts as the initiator: $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{uB}, 2)$, $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{uv}, 2)$
– In sessions where the party acts as the responder: $ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{uv}, 2)$, $ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{Au}, 2)$

Dealing with $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$'s computation of these values will be done in the following way:

– The other Diffie-Hellman private keys A, B are long-term private keys. $\mathcal{B}_{\mathsf{PRF\text{-}ODH}}$ can then use its own internal knowledge of the private keys to complete the computations.

Thus we have:

$$\mathsf{Adv}(break_3) \leq \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + \mathsf{Adv}(break_4)$$

Case C.3, Game 5: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \mathrm{KDF}(\widetilde{ck}, g^{Ab}, 2)$. Since, by **Game 4**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values ck, k_2 with uniformly random values $\widetilde{ck}, \widetilde{k_2}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_5)$$

Case C.3, Game 6: In this game we replace the function $\mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$ used to compute $k_{\mathtt{i}}, k_{\mathtt{r}} := \mathrm{KDF}(\widetilde{ck}, \epsilon, 2)$. Since, by **Game 5**, $\widetilde{ck}$ is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values $k_{\mathtt{i}}, k_{\mathtt{r}}$ with uniformly random values $\widetilde{k_{\mathtt{i}}}, \widetilde{k_{\mathtt{r}}}$. Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\mathsf{Adv}(break_5) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}} + \mathsf{Adv}(break_6)$$

Case C.3, Game 7: In this game, the challenger flips a bit $\bar{b}$ and uses $\bar{b}$ instead of $\pi_i^s.b_2$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 6** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 2$ (i.e. when using the key $\widetilde{k_1}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_1}$ is a uniformly random and independent value, and the bit $\bar{b}$ has the same distribution as $\pi_i^s.b_2$, this change is sound. Thus,

$$\mathsf{Adv}(break_6) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_7)$$

In **Case C.3, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 2$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.3, Game 8: In this game, the challenger flips a bit $\bar{b}'$ and uses $\bar{b}'$ instead of $\pi_i^s.b_3$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 7** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 3$ (i.e. when using the key $\widetilde{k_2}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the key $\widetilde{k_2}$ is a uniformly random and independent value, and the bit $\bar{b}'$ has the same distribution as $\pi_i^s.b_3$, this change is sound. Thus,

$$\mathsf{Adv}(break_7) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_8)$$

In **Case C.3, Game 8**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma = 3$ and thus $\mathcal{A}$ has no advantage in guessing this challenge bit nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$.

Case C.3, Game 9: In this game, the challenger flips a bit $\bar{b}*$ and uses $\bar{b}*$ instead of $\pi_i^s.b_4$ when responding to $\mathrm{Enc}(i, s, ad, m_0, m_1)$ and $\mathrm{Dec}(j, t, ad, c)$ queries from $\mathcal{A}$ when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\mathsf{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\mathsf{aead}}$ acts exactly as in **Game 8** except responding to $\mathrm{Enc}(n, h, m_0, m_1)$ or $\mathrm{Dec}(n, h, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 4$ (i.e. when using the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$). $\mathcal{B}_{\mathsf{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

An adversary capable of distinguishing this change can break the aead security of the AEAD scheme. Since the keys $\widetilde{k_{\mathtt{i}}}$, $\widetilde{k_{\mathtt{r}}}$ are uniformly random and independent values, and the bit $\bar{b}*$ has the same distribution as $\pi_i^s.b_4$, this change is sound. Thus,

$$\mathsf{Adv}(break_8) \leq \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} + \mathsf{Adv}(break_9)$$

In **Case C.3, Game 9**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\varsigma$, where $\varsigma \geq 2$ and thus $\mathcal{A}$ has no advantage in guessing these challenge bits (and the other challenge bit is not used during the experiment as $\pi_i^s.fr_1 = 0$) nor in causing π_i^s to set $\mathsf{win} \leftarrow 1$. Thus:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fACCE},\mathbf{CC.3}}_{\mathrm{XK},n_P,n_S,\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathrm{H},\mathcal{B}_{\mathsf{coll}}} + n_P^2 n_S^2 \cdot \Big(3 \cdot \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{B}_{\mathsf{aead}}} \\ + \mathsf{Adv}^{\mathsf{nn\text{-}PRF\text{-}ODH}}_{\mathrm{KDF},G,p,\mathcal{B}_{\mathsf{PRF\text{-}ODH}}} + 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{KDF},\mathcal{B}_{\mathsf{prf}}}\Big)\end{aligned}$$

F Full Patterns

In this section we give a full algorithmic description of the Noise Protocol Framework Patterns.

Initiator | **Responder**

$ck, h \leftarrow \mathrm{H}(\texttt{N\_label})$

$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$

$h \leftarrow \mathrm{H}(h \| g^B)$

$a \leftarrow_{\$} \mathbb{Z}_p$

$h \leftarrow \mathrm{H}(h \| g^a)$

$ck_0, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$

$c_0 \leftarrow \mathrm{Enc}(k_0, 0, h, m_0)$

$\xrightarrow{g^a, c_0}$

if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**

$h \leftarrow \mathrm{H}(h \| c_0)$

$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$

$\xRightarrow{\text{Payload Data}}$

Fig. 16: Noise Pattern $\texttt{N} : \leftarrow \texttt{s}, \ldots, \rightarrow \texttt{e}, \texttt{es}$.

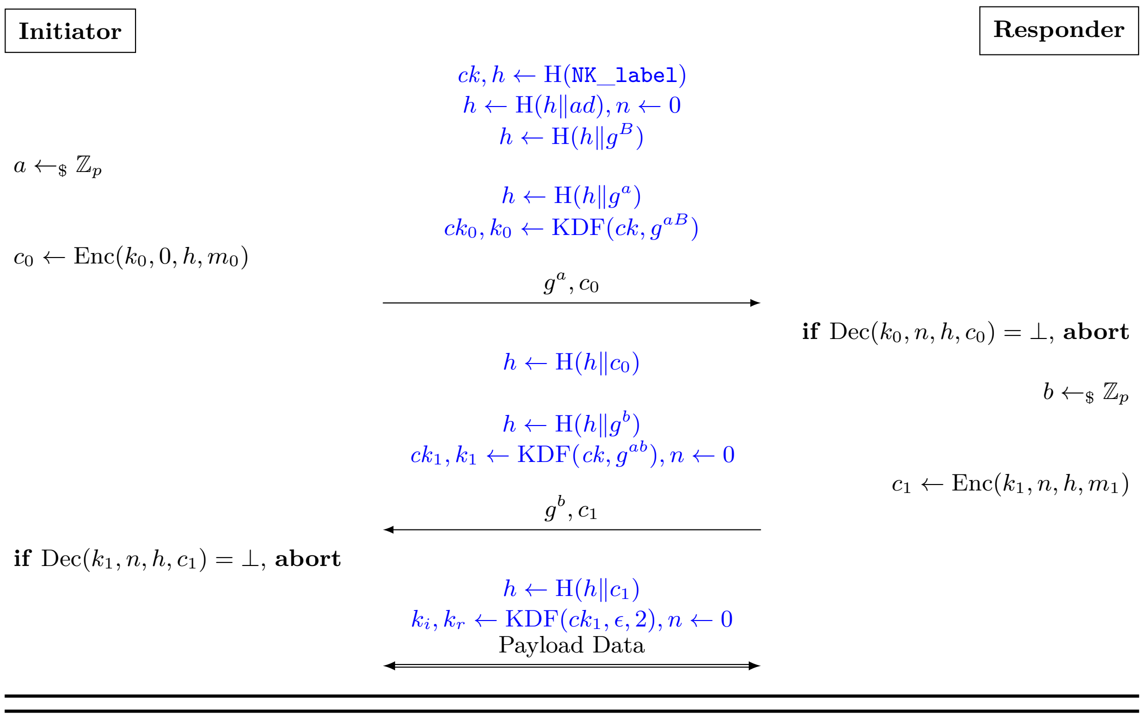

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{NK\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h \| g^B)$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$ck_0, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB})$
$c_0 \leftarrow \mathrm{Enc}(k_0, 0, h, m_0)$
g^a, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$b \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck_1, k_1 \leftarrow \mathrm{KDF}(ck, g^{ab}), n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_1)$
g^b, c_1
if $\mathrm{Dec}(k_1, n, h, c_1) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_1)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_1, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 17: Noise Pattern $\texttt{NK} : \leftarrow \texttt{s}, \ldots, \rightarrow \texttt{e}, \texttt{es}$.

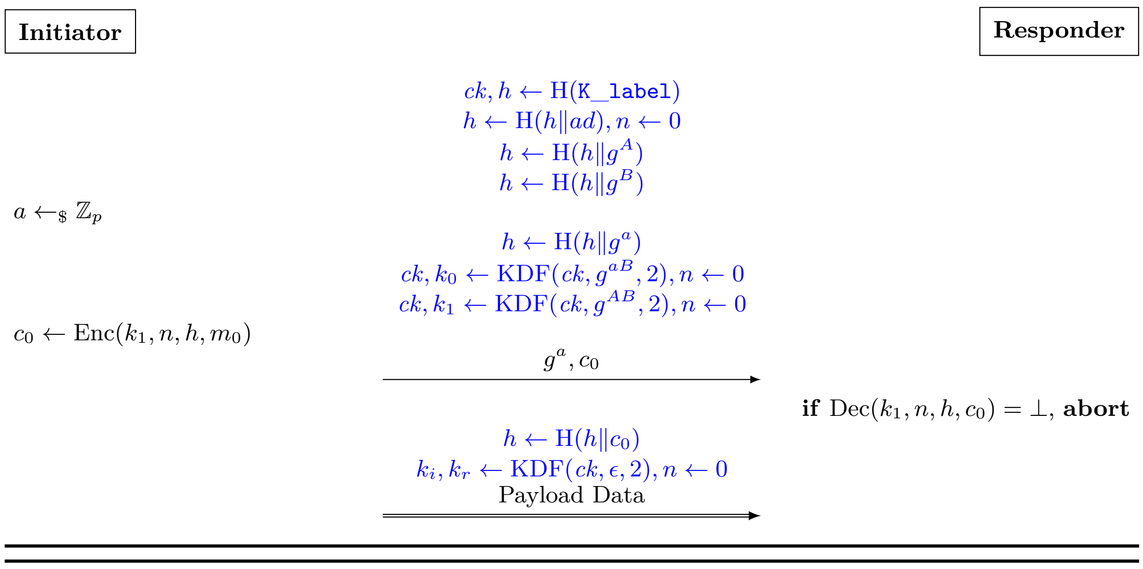

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{K\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h \| g^A)$
$h \leftarrow \mathrm{H}(h \| g^B)$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{AB}, 2), n \leftarrow 0$
$c_0 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$
g^a, c_0
if $\mathrm{Dec}(k_1, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 18: Noise Pattern $\texttt{K} : \rightarrow \texttt{s}, \leftarrow \texttt{s}, \ldots, \rightarrow \texttt{e}, \texttt{es}, \texttt{ss}$.

Initiator		**Responder**
	$ck, h \leftarrow \mathrm{H}(\texttt{KK\_label})$	
	$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$	
	$h \leftarrow \mathrm{H}(h \| g^A)$	
	$h \leftarrow \mathrm{H}(h \| g^B)$	
$a \leftarrow_\$ \mathbb{Z}_p$		
	$h \leftarrow \mathrm{H}(h \| g^a)$	
	$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$	
	$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{AB}, 2), n \leftarrow 0$	
$c_0 \leftarrow \mathrm{Enc}(k_1, 0, h, m_0)$		
	$\xrightarrow{g^a, c_0}$	
		if $\mathrm{Dec}(k_1, n, h, c_0) = \perp$, **abort**
	$h \leftarrow \mathrm{H}(h \| c_0)$	
		$b \leftarrow_\$ \mathbb{Z}_p$
	$h \leftarrow \mathrm{H}(h \| g^b)$	
	$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2), n \leftarrow 0$	
		$c_1 \leftarrow \mathrm{Enc}(k_2, n, h, m_1)$
	$\xleftarrow{g^b, c_1}$	
if $\mathrm{Dec}(k_2, n, h, c_1) = \perp$, **abort**		
	$h \leftarrow \mathrm{H}(h \| c_1)$	
	$k_i, k_r \leftarrow \mathrm{KDF}(ck_2, \epsilon, 2), n \leftarrow 0$	
	Payload Data $\Longleftrightarrow$	

Fig. 19: Noise Pattern KK $:\rightarrow$ s, $\leftarrow$ s, $\ldots, \rightarrow$ e, es, ss, $\leftarrow$ e, ee, se.

Initiator		**Responder**
	$ck, h \leftarrow \mathrm{H}(\texttt{X\_label})$	
	$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$	
	$h \leftarrow \mathrm{H}(h \| g^B)$	
$a \leftarrow_\$ \mathbb{Z}_p$		
	$h \leftarrow \mathrm{H}(h \| g^a)$	
	$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2)$	
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, g^A)$		
	$h \leftarrow \mathrm{H}(h \| c_0)$	
	$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{AB}, 2), n \leftarrow 0$	
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_1)$		
	$\xrightarrow{g^a, c_0, c_1}$	
		if $\mathrm{Dec}(k_0, n, h, c_0), \mathrm{Dec}(k_1, n, h, c_1) = \perp$ **abort**
	$h \leftarrow \mathrm{H}(h \| c_1)$	
	$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$	
	Payload Data $\Longrightarrow$	

Fig. 20: Noise Pattern X $:\leftarrow$ s, $\ldots, \rightarrow$ e, es, s, ss.

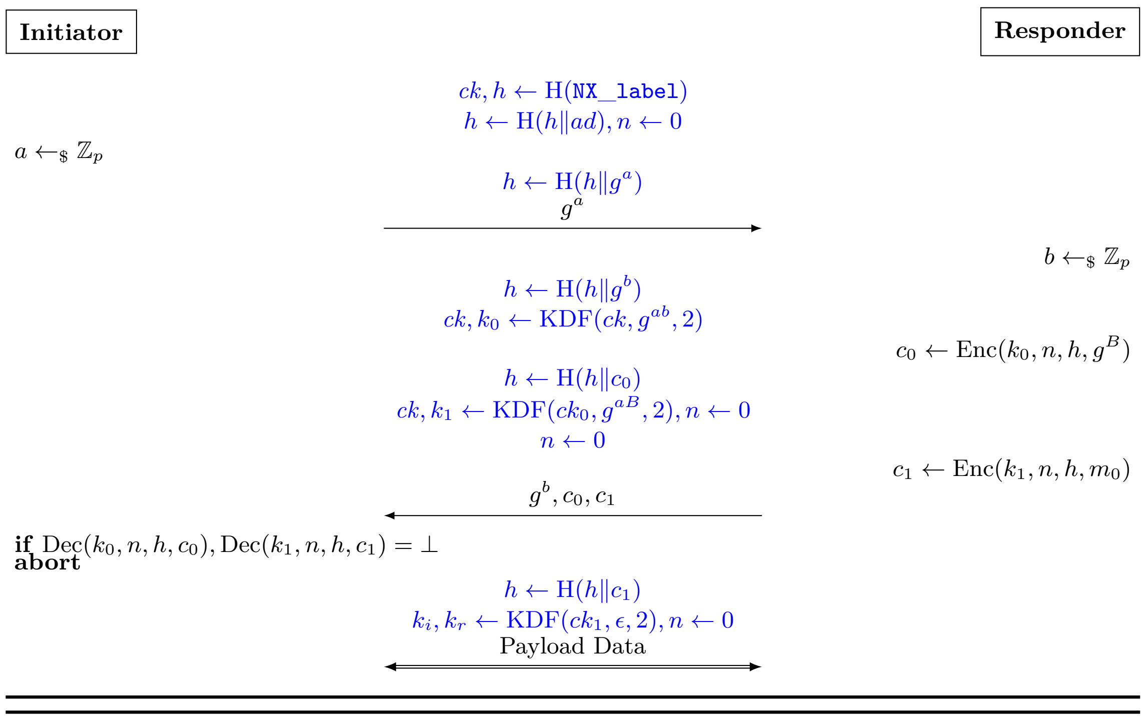

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{NX\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
g^a
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, g^B)$
$h \leftarrow \mathrm{H}(h \| c_0)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck_0, g^{aB}, 2), n \leftarrow 0$
$n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$
g^b, c_0, c_1
if $\mathrm{Dec}(k_0, n, h, c_0), \mathrm{Dec}(k_1, n, h, c_1) = \perp$
abort
$h \leftarrow \mathrm{H}(h \| c_1)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_1, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 21: Noise Pattern NX $:\rightarrow$ e, $\leftarrow$ e, ee, s, es.

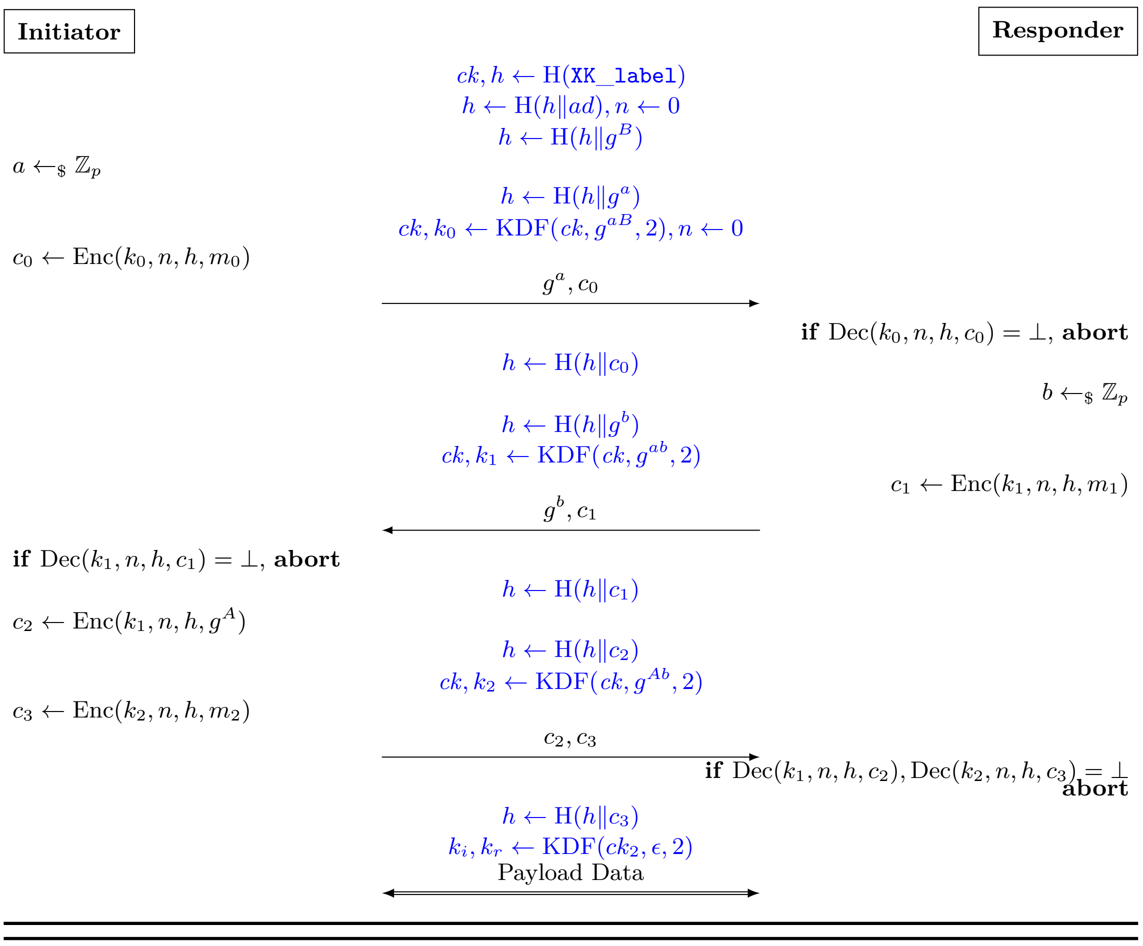

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{XK\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h \| g^B)$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, m_0)$
g^a, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$b \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_1)$
g^b, c_1
if $\mathrm{Dec}(k_1, n, h, c_1) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_1)$
$c_2 \leftarrow \mathrm{Enc}(k_1, n, h, g^A)$
$h \leftarrow \mathrm{H}(h \| c_2)$
$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2)$
$c_3 \leftarrow \mathrm{Enc}(k_2, n, h, m_2)$
c_2, c_3
if $\mathrm{Dec}(k_1, n, h, c_2), \mathrm{Dec}(k_2, n, h, c_3) = \perp$ **abort**
$h \leftarrow \mathrm{H}(h \| c_3)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_2, \epsilon, 2)$
Payload Data

Fig. 22: Noise Pattern $\texttt{XK} :\leftarrow \texttt{s}, \ldots, \rightarrow \texttt{e}, \texttt{es} \leftarrow \texttt{e}, \texttt{ee} \rightarrow \texttt{s}, \texttt{se}$.

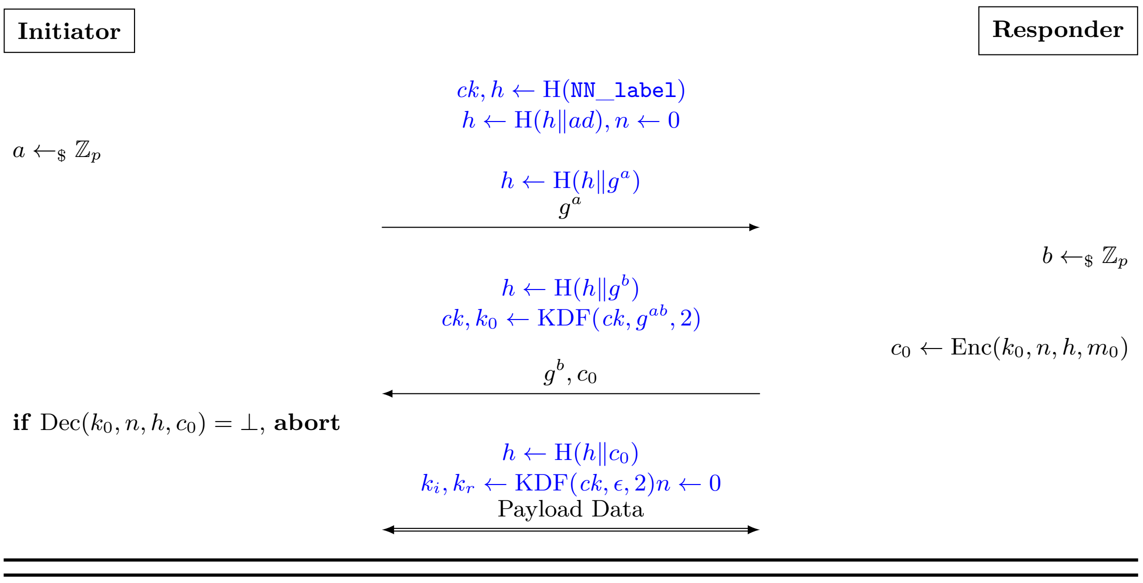

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{NN\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
g^a
$b \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, m_0)$
g^b, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2) n \leftarrow 0$
Payload Data

Fig. 23: Noise Pattern $\texttt{NN} :\rightarrow \texttt{e}, \leftarrow \texttt{e}, \texttt{ee}$.

Initiator | **Responder**

$ck, h \leftarrow \mathrm{H}(\texttt{KN\_label})$
$h \leftarrow \mathrm{H}(h\|ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h\|g^A)$

$a \leftarrow_{\$} \mathbb{Z}_p$

$h \leftarrow \mathrm{H}(h\|g^a)$

$$\xrightarrow{g^a}$$

$b \leftarrow_{\$} \mathbb{Z}_p$

$h \leftarrow \mathrm{H}(h\|g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2), n \leftarrow 0$

$c_0 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$

$$\xleftarrow{g^b, c_0}$$

if $\mathrm{Dec}(k_1, n, h, c_0) = \bot$, **abort**

$h \leftarrow \mathrm{H}(h\|c_0)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_1, \epsilon, 2), n \leftarrow 0$

$$\xleftrightarrow{\text{Payload Data}}$$

Fig. 24: Noise Pattern KN $:\rightarrow$ s, ..., $\rightarrow$ e, $\leftarrow$ e, ee, se.

Initiator | **Responder**

$ck, h \leftarrow \mathrm{H}(\texttt{KX\_label})$
$h \leftarrow \mathrm{H}(h\|ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h\|g^A)$

$a \leftarrow_{\$} \mathbb{Z}_p$

$h \leftarrow \mathrm{H}(h\|g^a)$

$$\xrightarrow{g^a}$$

$b \leftarrow_{\$} \mathbb{Z}_p$

$h \leftarrow \mathrm{H}(h\|g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2), n \leftarrow 0$

$c_0 \leftarrow \mathrm{Enc}(k_1, n, h, g^B)$

$h \leftarrow \mathrm{H}(h\|c_0)$
$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$

$c_1 \leftarrow \mathrm{Enc}(k_2, n, h, m_0)$

$$\xleftarrow{g^b, c_0, c_1}$$

if $\mathrm{Dec}(k_1, n, h, c_0), \mathrm{Dec}(k_2, n, h, c_1) = \bot$
abort

$h \leftarrow \mathrm{H}(h\|c_1)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$

$$\xleftrightarrow{\text{Payload Data}}$$

Fig. 25: Noise Pattern KX $:\rightarrow$ s, ..., $\rightarrow$ e, $\leftarrow$ e, ee, se, s, es.

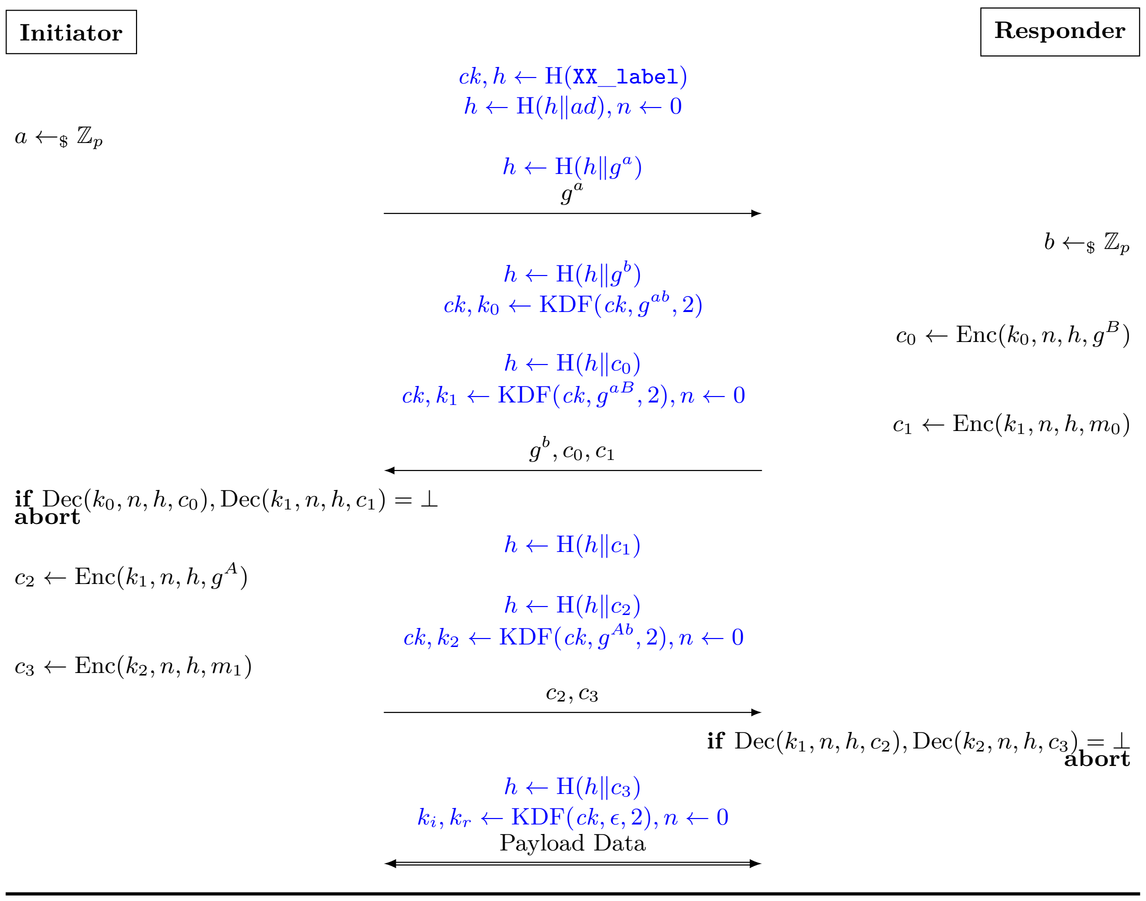

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{XX\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
g^a
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, g^B)$
$h \leftarrow \mathrm{H}(h \| c_0)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$
g^b, c_0, c_1
if $\mathrm{Dec}(k_0, n, h, c_0), \mathrm{Dec}(k_1, n, h, c_1) = \perp$
abort
$h \leftarrow \mathrm{H}(h \| c_1)$
$c_2 \leftarrow \mathrm{Enc}(k_1, n, h, g^A)$
$h \leftarrow \mathrm{H}(h \| c_2)$
$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2), n \leftarrow 0$
$c_3 \leftarrow \mathrm{Enc}(k_2, n, h, m_1)$
c_2, c_3
if $\mathrm{Dec}(k_1, n, h, c_2), \mathrm{Dec}(k_2, n, h, c_3) = \perp$
abort
$h \leftarrow \mathrm{H}(h \| c_3)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 26: Noise Pattern `XX` $:\rightarrow$ e, $\leftarrow$ e, ee, s, es, $\rightarrow$ s, se.

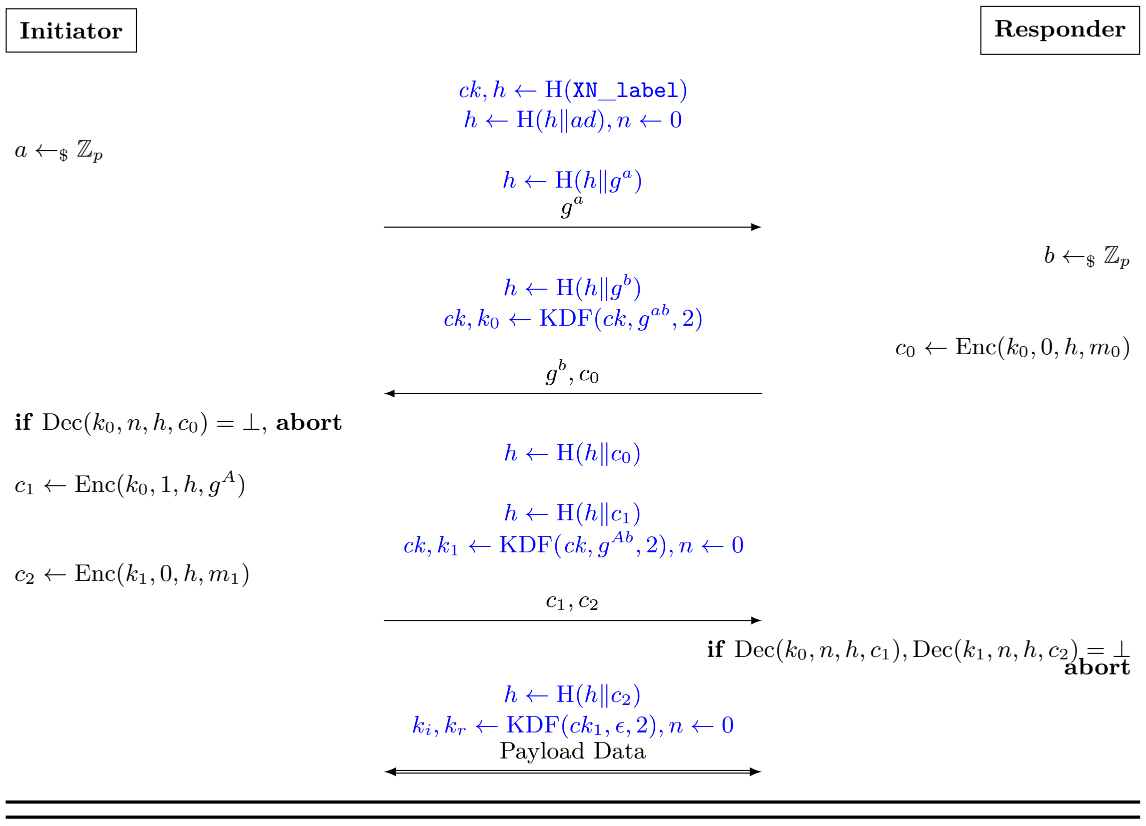

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{XN\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
g^a
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$c_0 \leftarrow \mathrm{Enc}(k_0, 0, h, m_0)$
g^b, c_0
if $\mathrm{Dec}(k_0, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$c_1 \leftarrow \mathrm{Enc}(k_0, 1, h, g^A)$
$h \leftarrow \mathrm{H}(h \| c_1)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2), n \leftarrow 0$
$c_2 \leftarrow \mathrm{Enc}(k_1, 0, h, m_1)$
c_1, c_2
if $\mathrm{Dec}(k_0, n, h, c_1), \mathrm{Dec}(k_1, n, h, c_2) = \perp$ **abort**
$h \leftarrow \mathrm{H}(h \| c_2)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck_1, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 27: Noise Pattern XN $:\rightarrow \mathsf{e}, \leftarrow \mathsf{e}, \mathsf{ee}, \rightarrow \mathsf{s}, \mathsf{se}$.

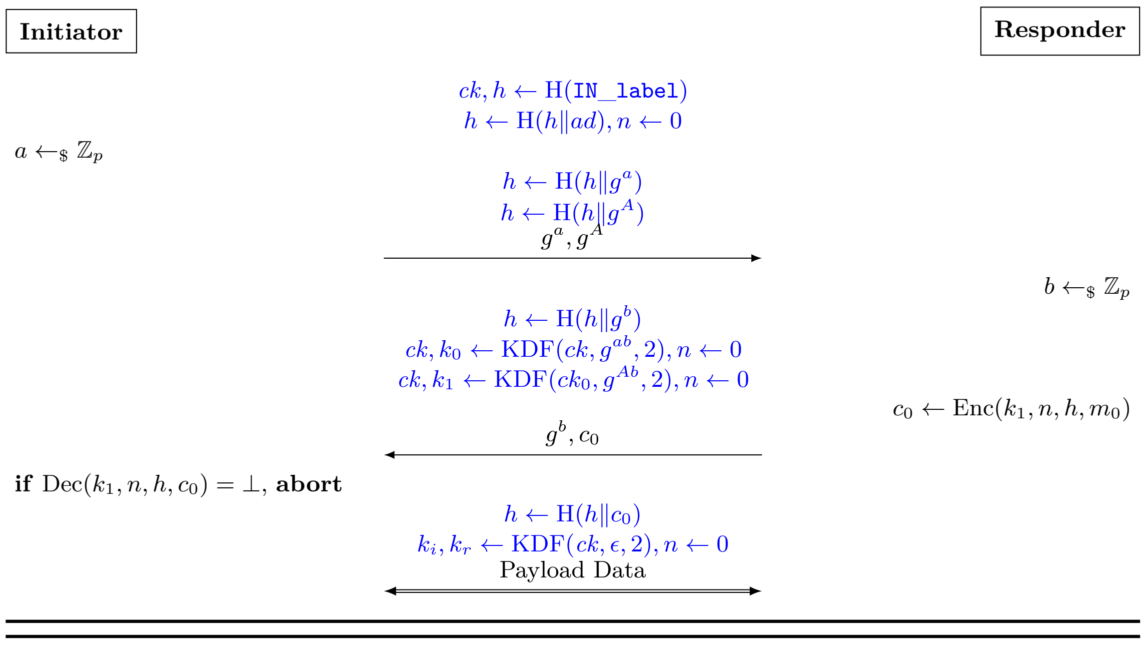
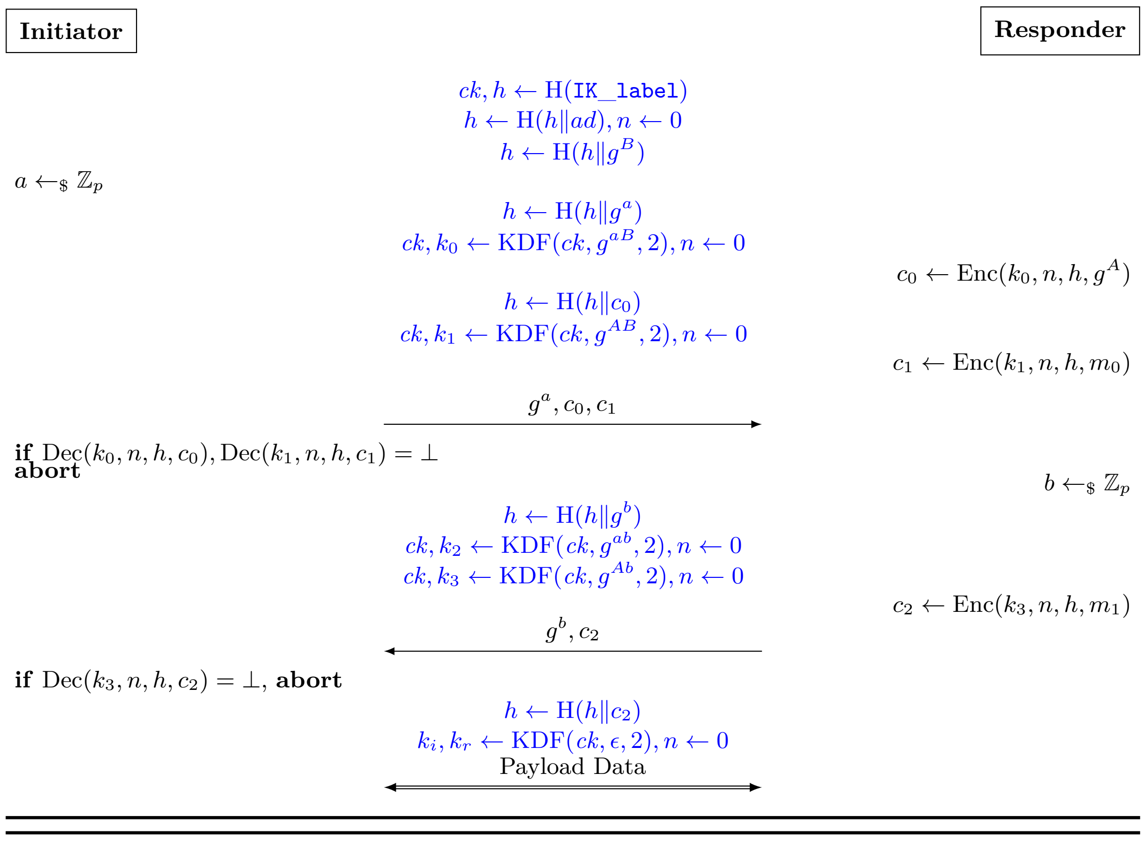
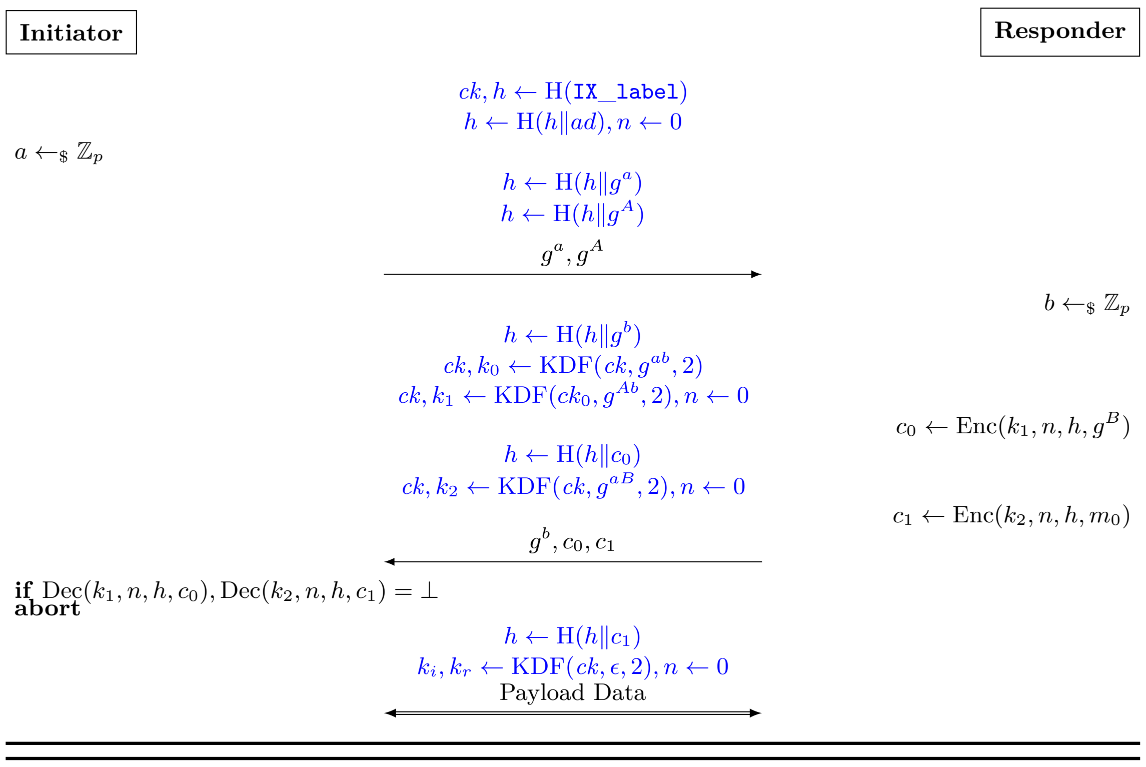

Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{IN\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$h \leftarrow \mathrm{H}(h \| g^A)$
g^a, g^A
$b \leftarrow_{\$} \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2), n \leftarrow 0$
$ck, k_1 \leftarrow \mathrm{KDF}(ck_0, g^{Ab}, 2), n \leftarrow 0$
$c_0 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$
g^b, c_0
if $\mathrm{Dec}(k_1, n, h, c_0) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_0)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 28: Noise Pattern IN $:\rightarrow \mathsf{e}, \mathsf{s}, \leftarrow \mathsf{e}, \mathsf{ee}, \mathsf{se}$.



Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{IK\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$h \leftarrow \mathrm{H}(h \| g^B)$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$c_0 \leftarrow \mathrm{Enc}(k_0, n, h, g^A)$
$h \leftarrow \mathrm{H}(h \| c_0)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck, g^{AB}, 2), n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_1, n, h, m_0)$
g^a, c_0, c_1
if $\mathrm{Dec}(k_0, n, h, c_0), \mathrm{Dec}(k_1, n, h, c_1) = \perp$ **abort**
$b \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2), n \leftarrow 0$
$ck, k_3 \leftarrow \mathrm{KDF}(ck, g^{Ab}, 2), n \leftarrow 0$
$c_2 \leftarrow \mathrm{Enc}(k_3, n, h, m_1)$
g^b, c_2
if $\mathrm{Dec}(k_3, n, h, c_2) = \perp$, **abort**
$h \leftarrow \mathrm{H}(h \| c_2)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 29: Noise Pattern $\texttt{IK} : \leftarrow \mathtt{s} \ldots \rightarrow \mathtt{e}, \mathtt{es}, \mathtt{s}, \mathtt{ss} \ldots \leftarrow \mathtt{e}, \mathtt{ee}, \mathtt{se}$.



Initiator
Responder
$ck, h \leftarrow \mathrm{H}(\texttt{IX\_label})$
$h \leftarrow \mathrm{H}(h \| ad), n \leftarrow 0$
$a \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^a)$
$h \leftarrow \mathrm{H}(h \| g^A)$
g^a, g^A
$b \leftarrow_\$ \mathbb{Z}_p$
$h \leftarrow \mathrm{H}(h \| g^b)$
$ck, k_0 \leftarrow \mathrm{KDF}(ck, g^{ab}, 2)$
$ck, k_1 \leftarrow \mathrm{KDF}(ck_0, g^{Ab}, 2), n \leftarrow 0$
$c_0 \leftarrow \mathrm{Enc}(k_1, n, h, g^B)$
$h \leftarrow \mathrm{H}(h \| c_0)$
$ck, k_2 \leftarrow \mathrm{KDF}(ck, g^{aB}, 2), n \leftarrow 0$
$c_1 \leftarrow \mathrm{Enc}(k_2, n, h, m_0)$
g^b, c_0, c_1
if $\mathrm{Dec}(k_1, n, h, c_0), \mathrm{Dec}(k_2, n, h, c_1) = \perp$ **abort**
$h \leftarrow \mathrm{H}(h \| c_1)$
$k_i, k_r \leftarrow \mathrm{KDF}(ck, \epsilon, 2), n \leftarrow 0$
Payload Data

Fig. 30: Noise Pattern $\texttt{IX} : \rightarrow \mathtt{e}, \mathtt{s}, \leftarrow \mathtt{e}, \mathtt{ee}, \mathtt{se}, \mathtt{s}, \mathtt{es}$.